University of Alberta

Multi-Project Software Engineering Management Using Systems Thinking

by

Bengee Lee

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science.

Department of Electrical and Computer Engineering

Edmonton, Alberta

Spring 2004

Library and Archives Canada
Published Heritage Branch
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada
Direction du Patrimoine de l'édition
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-96508-2
Our file Notre référence
ISBN: 0-612-96508-2

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Software organizations often have multiple projects developed concurrently. A multi-project environment is a complex and dynamic system. To analyze systematically, we need to delve into systems thinking. The dynamics models are divided into two categories: Shared models and Individual Project models. Shared models describe the relationships common to all projects. The Individual Project models include the interactions that are unique to each project. However, systems thinking alone lacks the capability to construct a multi-project scheduling network. Thus, this research is also investigating the integration of systems thinking with a multi-project network constructing method, called Critical Chain Project Management. Furthermore, in order to portray unexpected situations in project development, scenario planning is incorporated into the system dynamics models and the network. With such unique melding, the project manager can identify the restraining factors in various possible scenarios in a multi-project network, founded by systems thinking, to provide feasible management solutions.

Acknowledgment

This research has been a challenging but most worthwhile experience in the past few years. This would not have been possible without the support of many instructors, friends and family. First and foremost, I would like to express my deepest gratitude to my advisor, Professor James Miller. His invaluable expertise in Software Engineering along with his guidance and advise has helped me throughout this research. I would not have completed my Master's program without his generous financial and intellectual support. He has provided me unlimited amount of encouragement, suggestions and proper directions that have enabled constant improvement and refinement of this research.

I would also like to take this opportunity to acknowledge Dr. John D. Whittaker, professor of Engineering Management in the Department of Mechanical Engineering at the University of Alberta. I had a golden opportunity to take his Engineering Management course, i.e., Managing in a Technical Environment (ENGM 650) during the Fall term in 2002. Professor Whittaker is an excellent instructor, and has provided me invaluable insights and materials in the management field, which I have benefited from in this research.

I would like to express my appreciation to Dr. Petr Musilek, Dr. Whittaker and Dr. Miller for spending their invaluable time on reading and correcting this thesis, and for being the members of the examining committee.

Finally, I would like to thank my beloved parents, BonChin Lee and PohIn Oh for their love, nurture and support they have given me throughout my life, in good times and rough times. They have provided me the strength and courage to mount greater heights and pursue my dreams that finally lead to the completion of my research. I certainly would not have this opportunity in the University of Alberta without their understanding and unconditional love. Thank you.

Table of Contents

1.0 INTRODUCTION 1

2.0 BACKGROUND 8

2.1 PREVIOUS METHODOLOGY 8

2.1.1 PERT/CPM 8

2.1.2 Multi-Channel Queuing System 9

2.1.3 Abdel-Hamid's Dynamic Models 10

2.2 MANAGEMENT APPROACH 12

2.2.1 Common Approach 12

2.2.2 Customized Approach 13

2.2.3 Hybrid Approach 14

2.3 SYSTEMS THINKING 15

2.3.1 Causal Loop Diagram 17

2.3.2 System Dynamics 18

2.4 SCENARIO MODEL 21

3.0 MULTI-PROJECT MANAGEMENT MODELS 24

3.1 MULTI-PROJECT NETWORK 24

3.1.1 Step 1 - Development of Project Network 25

3.1.2 Step 2 - Resource Leveling 27

3.1.3 Step 3 - Development of the Critical Chain 30

3.1.4 Step 4 - Placement of Buffers 31

3.2 SYSTEM DYNAMICS MODELS 33

3.2.1 Shared Models 36

3.2.1.1 Workload and Exhaustion Model 36

3.2.1.2 Human Resource Model 45

3.2.2 Individual Project Models 55

3.2.2.1 Effort Model 55

3.2.2.2 Productivity Model 57

3.2.2.3 Control Model 66

3.2.2.4 Planning Model 72
3.3 SCENARIO MODELING 85
3.3.1 Event Scenario 86
3.3.2 Policy Scenario 88
3.3.3 Theory Scenario 89
3.3.4 Strategy Scenario 91
3.3.5 Application of Scenario Modeling 92
3.3.5.1 Event Element vs. Event Element 93
3.3.5.2 Policy Element vs. Event Element 96
3.3.5.3 Strategy Element vs. Policy Element 97
3.3.5.4 Theory Element vs. Policy Element 100
4.0 MODEL INTEGRATION 103
4.1 EXAMPLES 105
4.1.1 Scenario 1 Modeling 107
4.1.2 Scenario 2 Modeling 111
4.1.3 Scenario 3 Modeling 114
5.0 FUTURE RESEARCH 120
5.1 MODEL SIMULATION 120
5.2 RESOURCE ALLOCATION HEURISTICS 121
6.0 CONCLUSIONS 124
BIBLIOGRAPHY 127
APPENDIX 132

List of Tables

Table 3.1: Project I and II Properties 25
Table 3.2: Tendency of quitting 42
Table 3.3: Effect of unemployment rate on hiring 52
Table 3.4: Event scenario 88
Table 3.5: Policy scenario 89
Table 3.6: Strategy scenario 92
Table 4.1: Project Properties 106
Table 4.2: Scenario 1 Results 111
Table 4.3: Scenario 2 Results 113
Table 4.4: Scenario 3 Criteria and Assumptions 114
Table 4.5: Quitting rate due to inability to cope 115

List of Figures

Figure 1.1: Overall View of A Multi-Project Management Simulation Model 6
Figure 2.1: Causal loop diagram 18
Figure 2.2: Flow Diagram 21
Figure 3.1: Development of Project Network 27
Figure 3.2: Modified Wiest and Levy Resource Leveling Heuristics 29
Figure 3.3: Resource Leveling 30
Figure 3.4: Development of critical chain and buffer placement 32
Figure 3.5: A Simplified View of Multi-project Subsystem Models 35
Figure 3.6: Workload and Exhaustion Model in causal-loop diagram 37
Figure 3.7: Workload and Exhaustion Model in flow diagram 38
Figure 3.8: Ability to cope 40
Figure 3.9: Rate of increase in exhaustion level 44
Figure 3.10: Human Resource Model (General) in causal-loop diagram 47
Figure 3.11: Human Resource Model (General) in flow diagram 47
Figure 3.12: Human Resource Model (Extended) in causal-loop diagram 50
Figure 3.13: Human Resource Model (Extended) in flow diagram 51
Figure 3.14: Effort Model in causal-loop diagram 57
Figure 3.15: Effort Model in flow diagram 57
Figure 3.16: Productivity Model in causal-loop diagram 59
Figure 3.17: Productivity Model in flow diagram 60
Figure 3.18: Communication Overhead 61
Figure 3.19: Increase of staff's ability due to training 63
Figure 3.20: Control Model (General) in causal-loop diagram 67
Figure 3.21: Control Model (General) in flow diagram 68
Figure 3.22: Control Model (Extended) in causal-loop diagram 71
Figure 3.23: Control Model (Extended) in flow diagram 72
Figure 3.24: Planning Model (General) in causal-loop diagram 74
Figure 3.25: Planning Model (General) in flow diagram 75

Figure 3.26: Planning Model (Extended) in causal-loop diagram 77
Figure 3.27: Planning Model (Extended) in flow diagram 78
Figure 3.28: General form of WCWF-1 80
Figure 3.29: General form of WCWF-2 82
Figure 3.30: Behavior of schedule adjustment 84
Figure 3.31: Modified form of WCWF-1 90
Figure 3.32: Example of scenario quadrants 93
Figure 3.33: Requirement volatility vs. Maximum tolerable project duration 94
Figure 3.34: Average HR cost per employee vs. Number of projects 96
Figure 3.35: Project end date vs. Ceiling on target workforce 98
Figure 3.36: Willingness to hire late in development vs. Experienced staff over new staff ratio 100
Figure 4.1: A Simplified View of Multi-Project Management Model 104
Figure 4.2: CCPM network before size increment 107
Figure 4.3: Gantt chart representation before size increment 108
Figure 4.5: CCPM network after 40% size increment in Project II 109
Figure 4.6: Gantt chart representation after 40% size increment in Project II 110
Figure 4.8: Combined system dynamics model with 40% size increment 112
Figure 4.9: Cumulative Tasks Developed & Required Tasks 115
Figure 4.10: Exhaustion Level & Overwork Duration Threshold 116
Figure 4.11: Workload, Ability to cope, Actual fraction of a man-day & Number of employees 117

1.0 INTRODUCTION

Software organizations are fundamentally multi-project oriented in order to remain competitive in industry by reducing project lead-time, cost and resource consumption while improving quality, product functionality and application domain. The growth rate of parallel, multiple projects in need of program management is phenomenal, between 20% and 30% a year (Dye 2000). The imperative is forcing organizations to rethink how they implement multiple software projects from the ground up. However, unlike traditional project management analysis that handles the projects individually, contemporary project management needs to involve the interdependencies of various concurrent projects that are of different sizes, productivities, resource requirements from a common pool, and progress levels (Abdel-Hamid 1991). Managing a single project, without considering its influence it may have on other projects, may not be beneficial to its organization and may actually degrade the entire performance of the organization. The behavior of the underlying dynamics of multi-projects is not fully explored and these interdependencies must be recognized which lie in the structure of a portfolio of projects.

Before dwelling on the management issues, we need a proper understanding of the workings of some basic elements. Investigations into these planning and control functions have found several fundamental characteristics in the multi-project environment (Walker 2000):

- Multiple projects are interdependent due to the use of common resources.
- There are complex trade-offs between the utilization of resources and the on-time completion of individual projects.
- Some methods must be employed to prioritize the use of resources among multiple projects.
- A control mechanism must exist to reduce the variance between planned and actual project completion dates.

Based on those fundamental characteristics, we can anticipate several issues that affect multi-project management more significantly than single project management:

- Resource allocations: How to maximize the utilization of resources across various projects? Human resources, in particular, are scarce and valuable in the organization, and some efficiency and schedule problems may exist when some resources are overloaded with work, while others are under-utilized. In times of schedule urgency, the management needs to consider when to increase the workforce in the project to cope with the timeline.
- Multi-project scheduling: How to anticipate a delay in a single project may affect the schedule of other projects? As several projects coexist in the organization, and constantly demand the use of common resources, a change in schedule in one project may have a propagating effect on others. Due to the restricting nature of resource availability, the management may need to consider alternatives and trade-offs to compensate and minimize the impact of one project on other remaining projects.
- Project priorities: How to distribute resources among multiple projects based on project priorities? Ignoring project priority as one of the factors when making resource allocation may have a devastating effect on customer satisfaction, product quality and complete time. Projects with high priority should draw more attention and resources than those with lower priority, and top management should have a definite guideline or heuristics to allocate resources based on those criteria.
- Project methodology: How to incorporate several independent project methodologies in a multi-project environment? Several useful project management methodologies are common, such as PERT (Program Evaluation and Review Technique), Gantt chart, earned value management, system dynamics, risk management, and some proprietary approaches developed by consulting firms, just to name a few. These approaches are mostly used in an independent manner without blending into one coherent methodology. For example, PERT chart is used for schedule planning, and risk management is

used for accessing possible risks and solutions, while system dynamics is used to dynamically model the project environment. These methodologies have their individual unique strengths and weaknesses, and when they are used separately in managing multiple projects, the outcome may not be as successful and effective when the strengths from various methodologies are combined together by integrating those approaches into one.

- Unplanned scenarios: How to anticipate and plan for the unknowns? If everything were to follow the plans exactly, there would be no failure in project development. Unknowns and unplanned cases always happen in most projects. Due to the interdependencies of multi-project environment, the effect of those unforeseen scenarios in one project may impact other projects simultaneously. For instance, if a significant amount of employees resigned suddenly in the middle of a project development, the impact of this scenario is definitely affecting the entire organization. Although it is difficult to avoid the unknowns totally, it is possible to plan and react when such occasions occur in future.
- Exhaustion: How to understand the workforce exhaustion level in an environment where several difficult and easy projects coexist? Due to the scarcity of the resources, the workforce is constantly multi-tasking in several projects, and one of the physical and mental effects on the workforce is exhaustion. Undermining this effect may lead to high attrition rate and missed schedule. Hence, there exists a need to monitor this exhaustion level to prevent loss of valuable workforce.
- Top level visibility: How to see the entire network of projects simultaneously so that a decision or strategy made on one project may have little impact on others? Very few people seem to know or understand how different simultaneous projects fit into the work structure, let alone implement or support the organization's vision and strategy. Hence, the top management should posses this vision or means of studying the portfolio of projects concurrently to effectively plan and control the organization.

- Common language: How to communicate all development teams using a common language? Communication always tends to be a science of its own, as each development team may have their own unique way of presenting their plans, design, problems and solutions. The result leads to misalignment in top-level management strategy, as each project team expresses the progress and decision differently. Hence, this communication gap needs to be closed by standardizing a common language across the entire organization.

Clearly, multiple projects are inherently complex in nature due to these interdependencies and issues, and we believe that the ability to manage this kind of complexity determines the success or failure of an organization. One study found that 50% of the finished software projects exceeded budget by 60-190%, while only 25% were completed on time, within budget (Mahaney 2003). The reasons the projects were late are due to lack of resources and contention of common resources among various projects (Linberg 1999). This is just one of the complications of interdependency. In order to systematically analyze these mutual connections, we need to delve into systems thinking to find the solutions. The purpose is to assist us in systems thinking that enables us to see the complexity of the real world in a systematic way. By pursuing this approach, our goal is to achieve the following accomplishments:

- An understanding of the intricate interdependencies of concurrent projects. These dependencies may not be obvious without proper comprehension of the complex system. Section 2.3 will provide a guideline of using systems thinking to achieve the insights of a web of connectedness in a complex environment.
- An ability to plan and control, and thus manage the environment by considering various possible outcomes. Management depends on effective planning and control of the projects. The ability to foresee different outcomes, as a result of different management strategies, is also crucial to a multi-project organization.

- An ability to analyze different scenarios to achieve higher work efficiency and to avoid detrimental project failures. Scenario planning relies on the understanding of the system and studies the impact of different driving forces affecting the future. Its purpose is not to predict future, but to show how different cases can manipulate the future in different directions.
- A means to discuss the interdependencies and complexities with the management in a systematic and coherent way. Miscommunication and mismanagement may lie with the problems of conveying a complex system through linguistics, especially when the system involves multiple concurrent projects that have their individual characteristics and interdependencies. Our goal is thus to obtain a way of representing the system with diagrams, so that everyone has a universal and consistent way of communication. This is further discussed in Section 2.3.
- A foundation for using system dynamics to simulate the environment. Simulation provides experimentations with the outcomes of this research to assess the impact of different actions, options, and factors on the environment. This research caters the necessary foundation models to simulate the environment. System dynamics will be introduced in Section 2.4.

In general, we are able to make better, more robust and wiser decisions with systems thinking, since we are considering the problem by understanding the full consequences of each feasible solution. However, it does not provide a complete representation of a network of multi-projects in terms of schedule and resource allocation. Systems thinking alone may not fully capture the effect of schedule and resource allocation policies across the network, such as a schedule delay in one project may affect the lead-time in another project as they may share a common human resource in the project development. Each individual project may have its own unique set of development conditions and thus its own set of system dynamics models. For example, some projects may require fewer workforces due to the minimalism in size and complexity, and thus the models in each project are unique and separate. In order to study the interdependencies across multiple projects in terms

of schedule and resources, we need an inter-link between these models. Hence, we believe that the dynamics models for a multi-project organization should be built upon a shared foundation of a multi-project network that can accommodate the dynamics and interdependencies of various concurrent projects. The shared platform can be constructed using Critical Chain Project Management (CCPM) methodology (Goldratt 1997) that can be applied on the formation of a multi-project network. In addition, a Scenario model will be used to test different kinds of impact on the project behavior (Barros 2000). Different scenarios may arise in the project development, and a manager may study the influence of different management decisions on the projects using this model. The following diagram (Figure 1.1) provides an overall view of the multi-project management model that will be employed in this research. The functions of each model and the controller will be described further in their individual sections.

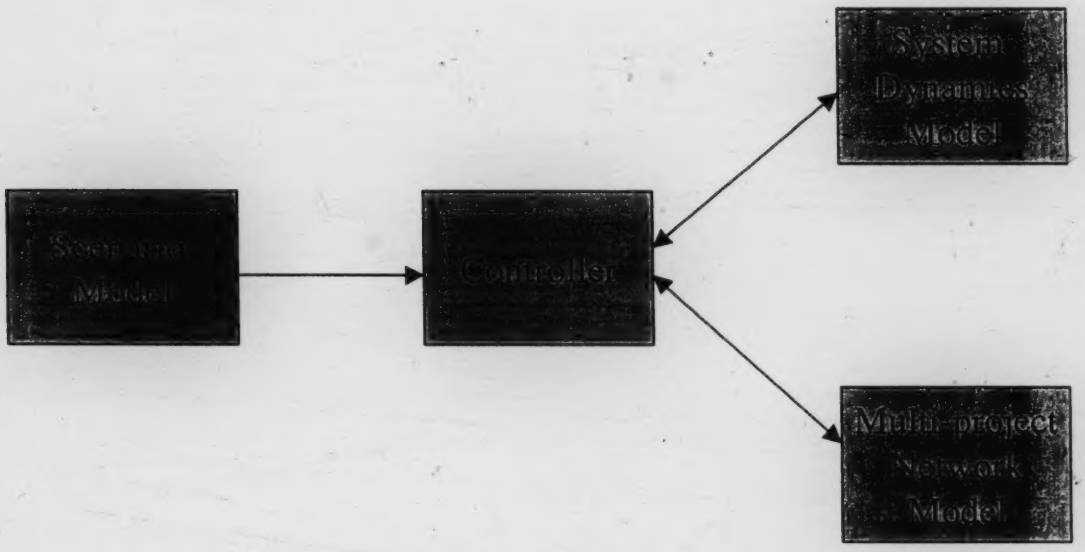


Figure 1.1: Overall View of A Multi-Project Management Simulation Model

This paper is organized in the following manner. The first section introduces the intention of this paper. In the second section, a few of the previous common management methodologies are introduced and their drawbacks are examined. In addition, a brief analysis of different management approaches practiced in a multi-

project environment is discussed, followed by the general principles of systems thinking and system dynamics with appropriate diagrams. Then, scenario modeling is introduced in the same section with its four major categories. In Section 3, various management methods are elaborated. The multi-project network is discussed in this section along with various steps required to form an integrated network. This section also deals in depth with system dynamics and related models used in this research, and then scenario modeling is explained with specific elements found in system dynamics models. The integration of system dynamics models, scenario models and multi-project network is addressed in Section 4. Next, some proposals for further research are provided in Section 5, and finally a conclusion to sum up the modeling concept is drawn in Section 6.

2.0 BACKGROUND

To provide certain basic understanding about the background of this research, the previous methodologies related to this field are discussed along with their strengths and weaknesses. In addition, various management approaches are explained along with the general knowledge of systems thinking and scenario modeling.

2.1 PREVIOUS METHODOLOGY

The project management field in general has grown significantly since the development of the tools of project management. This section discusses the various methodologies that have been developed and used in software engineering industry. However, each project management method has its own disadvantages that lead to this research development in the quest of creating a better way of managing multi-project.

2.1.1 PERT/CPM

PERT stands for Program Evaluation and Review Technique, and was developed by the U.S. Department of Navy in 1958 to manage the Polaris submarine missile program (Carayannis 2003). A similar methodology, the Critical Path Method (CPM) was developed for the private sector for project management at about the same time, and has become synonymous with PERT. Hence, the technique is commonly known as PERT/CPM.

PERT/CPM is a network-based method designed to assist in evaluating project scheduling and performance. It provides graphical representation of project activity networks and consists of some numbered nodes representing events, which are linked by labeled vectors representing activities in the project. As some activities in the network can be carried out in parallel, the project completion depends on the existence of several paths in a project. The critical path is the longest duration path throughout the network. The significance of the critical path is that any activity that lies on it cannot be delayed without affecting the entire project schedule. Due to its

impact on the entire project, the analysis of critical path is an important aspect of project planning.

Nevertheless, PERT/CPM alone without incorporating supplemental heuristics is criticized for its focus on single project environment, and Walker (2000) has listed out those limitations, gathered from various sources. PERT/CPM assumes path independence and does not recognize the variance on one path that may cause another path to be "late". Hence, any variability will cause the project duration to exceed the estimates and as the variability increases so does the difference between planned and actual project duration. The other disadvantage of PERT/CPM is its limitations in multi-tasking consideration. Multi-tasking is common in an environment with limited resources and multiple projects. Inefficiency in allocating concurrent tasks may yield inaccurate project planning.

To overcome these limitations in project management, this research recognizes the resource constraints while developing the project networks. Resource capacity is not infinite, as additional critical element is identified when the resources are not available. The variability of activity duration is allowed in this research work by stating the range of activity duration. The problem of multi-tasking is solved by introducing the fraction of effort on each project. The details of the solutions will be covered later when the methodology used in this research is introduced.

2.1.2 Multi-Channel Queuing System

Fatemi-Ghomi (2002) considered the multi-project resource allocation problem using a multi-channel queuing system. A framework was developed and a programming procedure was used to generate a simulation model using a general-purpose simulation language like GPSS. The model is used to investigate behavior of the completion-time random variable, as a function of the number of resources assigned to work on each project. The queuing system assumes the resource utilization is constant and does not fluctuate during the project development, and all projects use a

common resource. Constraint resource is then considered as the maximum number of channels in a queuing system. Activity is not performed until a channel is available.

The major limitation of this approach involves the assumption of constant resource utilization. In reality, the resource consumption level may fluctuate during the project development, and may have ripple effect on the multi-project network. In addition, the system is incapable of identifying critical paths in the network. Hence, the longest path due to schedule and resource constraints cannot be identified for project duration estimation. In terms of design flexibility, the programming procedure needs to be customized for any changes in the project network. In other words, the procedure is written based on a fixed and pre-determined project network, and for example, if one activity is to be added into the network, the entire procedure needs to be re-written.

Due to these limitations, this research does not employ the multi-channel queuing system in the model. Instead, a system dynamics model is used to allocate resources that may vary throughout the project development. Furthermore, CCPM (Critical chain project management) that is similar to PERT/CPM is used in the research to recognize the critical paths due to schedule and resource constraints and to provide flexibility in the project network for any activity changes.

2.1.3 Abdel-Hamid's Dynamic Models

In previous applications of research, advanced modeling techniques have been used to simulate the dynamic nature of systems thinking in software engineering. Abdel-Hamid provided a comprehensive treatment of the complexity encountered in the software development process, and a means of studying the dynamic interdependencies in the systems (Abdel-Hamid 1991). Systems thinking and system dynamics will be described in details in Section 2.3. The studies involved a comprehensive integrative model of software development management, divided into four major areas of software project management: human resource management, software production, project control, and project planning. Since then, the system dynamics models have been used extensively in the software project management

discipline, such as in education and training (Pfahl 2000) (Merrill 1997), and software processes modeling (Madachy 1996) (Lehman 2001).

Abdel-Hamid's work evolved around single project system and thus resource constraints were not considered. With no other projects to share the resources, the critical path of resource constraints was not defined. Hence, the resources were assumed to be always available when needed. In addition, the sequence of project activities was not clearly described and portrayed in the models. In fact, the sequence is only implied in the inter-relationships of the models. It lacks a comprehensive means of displaying the project activities, and as a result, the critical path of schedule constraints cannot be identified. Furthermore, project manager needs a thorough understanding of the models in order to study scenario planning. Normally, the study of scenario cases only involves certain manipulation of project parameters, and Abdel-Hamid's work does not clearly categorize and segregate those parameters from the complex models for the sake of abstraction. In the productivity model, Abdel-Hamid did not mention the effect of training on productivity improvement, as well as the impact of organization's profits on the human resource policies. Moreover, project size can vary when the complexity and/or the requirement change, and this fact is not discussed in his model. Beyond that, there are some other less critical limitations that are not mentioned here, but will later be introduced in Section 3.2.

Despite the limitations, Abdel-Hamid provides a solid foundation to study the dynamics of software engineering, and this research is based on Abdel-Hamid's work on system dynamics, particularly on the human resource, productivity, control and planning models. As multi-project environment is involved in this study, resource allocation and constraints have to be recognized and resolved, using both systems thinking and CCPM. When CCPM is employed, the activity sequence and critical path due to both resource and schedule constraints can be clearly determined, and thus it solves one of the limitations of Abdel-Hamid's models. A separate scenario model is created to study the scenario planning, and so the project manager does not need to possess the detailed and complex knowledge of the models in order to study

scenario cases. As for the limitations in the productivity, human resource and other models, new relationships are added into the models. Without providing further details of the research in this section, the remaining of this thesis offers a detailed description of the modified Abdel-Hamid's models, along with additional models incorporated into a multi-project domain.

2.2 Management Approach

Before going into the details of individual models, understanding management approach is crucial in a multi-project environment as it has been known that whenever an organization is undertaking a multi-project portfolio, it is likely the organization may take a common approach to the management of all projects, using the same reporting format, same resource allocation method, same management approach, etc. However, it has been proven in the research (Payne 1995) that the organizations that tailor the approach to the size of the project, and the resource type working on the project are more likely to achieve their project objectives, than those that treat all factors equal in weight and priority. The following analysis provides a brief understanding of the various management approaches.

2.2.1 Common Approach

It is a common practice in software organizations to use a single project management approach for all projects within the organization, regardless of the type of project, its size, or the type of resource used. The advantages are listed as below (Turner 1997):

- A universal consistent reporting mechanism can be adopted to give comparable progress reports across all projects;
- Resource requirements can be managed on a consistent basis to facilitate the management of capacity constraints;
- Human resource transfer flexibility between projects without having to relearn the management approach used project by project; and
- Small projects can be used as a training ground by new managers for future projects.

A natural assumption behind this approach is that the projects involved are fundamentally homogeneous and do not exhibit much diversified characteristics. However, adopting this approach solely without regarding the obvious complexity and variety of software projects may not be tactically wise and the project development teams may lose their flexibility to improvise when facing cost, resource and schedule constraints.

2.2.2 Customized Approach

Hence, despite the advantages, Payne (Payne 1995) discovered that organizations which used common approach regardless of project size, complexity, type and skill type, reported less success than organizations where people customized their project management approach. It is discovered that different project sizes have different kinds of emphasis on project management (Turner 1997). The project size can be categorized into 4 types: small, medium, large and major. A major project is roughly equal to the capitalization of the parent organization. A large project is one tenth smaller than the major project, a medium project is one tenth smaller than the large project, and a small project is one tenth smaller than the medium project. The emphasis of each project size can be explained as follows:

- For small to medium sized projects, the main emphasis is on the prioritization of resources across several projects, and to avoid the bureaucracy of procedures designed for larger, more complex projects.
- For large projects, the main emphasis is on the coordination of a complex sequence of activities and balancing resources across the activities. Large projects also have greater demand of data management, compared to the small and medium sized projects. Large projects seem to suffer more than the small to medium projects when the common project management approach is used, perhaps indicating that all their management requirements were not met.
- For major projects, the main emphasis is on the coordination of the resource activities across several sub-projects, and managing the risk.

2.2.3 Hybrid Approach

Since both approaches have advantages, another approach has been developed that has these two combined features - a hybrid approach. Basically, some levels of the common approach are retained and some customizations are carried out at some levels. We believe that the advantages of the hybrid approach over the common and the customized approaches are:

- To provide flexibility and non-restrictive nature to accommodate the common features as well as the distinct features of the multi-projects.
- To enhance the efficiency of the tactical management by using a common basis for the appropriate features, such as documentation, and using a customized approach when necessary, such as scenario modeling.
- To ease of project scheduling by first planning each project on a common basis and then adjusting the individual project based on different priorities and requirements.
- To create different scenarios to study the tactical management for each project during simulations, by using a basic template for the scenarios but different settings for different projects.

The project models outlined in this research use the hybrid approach to provide a common foundation for similar parameters, and use the customized approach to deal with different unexpected scenarios. CCPM methodology creates a universal platform for a multi-project network that describes the project activity sequence, schedule and resource allocation. In system dynamics models, there are models that possess similar management strategies and interests, and these common models are shared among multiple projects. For those models that exhibit unique characteristics are separated from the influence of non-related projects and created as individual models. Scenario models are created with a basic pattern but allow the project managers to manipulate the parameters to suit individual scenario needs.

2.3 Systems Thinking

After introducing some general insights of various management approaches, the main concept in managing concurrent projects involves systems thinking. Systems thinking is the combination of an approach to problem solving and a set of tools, techniques, and methods that equip us with just what we are looking for: an appropriate toolkit for understanding complex systems and their associated properties (Sherwood 2002). Since Ackoff and Churchman introduced systems thinking in the field of organizational/management science (Churchman 1957), various fields including software engineering have adapted systems thinking idea to understand a complex system. It is required that we change our way of thinking and analysis. We need to move away from looking at isolated events and their causes, and start to look at the entire system that is made up of interacting parts (Kirkwood 1998). For instance, in the context of this paper, we should not look at individual projects and their elements separately, but we need to understand the entire organization that is running several concurrent projects and the project interactions within the organization. The interactions may include human resource that is shared among the projects, overwork and exhaustion that are caused by multi-tasking, and the effect of staff retention and attrition on the allocation of manpower across various projects. In other words, systems thinking is also a study of the collective behavior of its components and the connectedness between the components that comprise the system of interest. As nicely phrased by Sherwood (Sherwood 2002):

- If you wish to understand a system, and so be in a position to predict its behavior, it is necessary to study the system as a whole. Cutting it up into bits for study is likely to destroy the system's connectedness, and hence the system itself.
- If you wish to influence or control the behavior of a system, you must act on the system as a whole. Tweaking it in one place in the hope that nothing will happen in another is doomed to failure – that's what connectedness is all about.

To explain this connectedness and collective behavior in more detail, an example from a simulation modeling of a software process is used (Donzelli 2001). Over a software project development time, the requirements grow from an initial amount of 1500 function points (FP) to an amount of 1500 FP+20%, and in addition to the 15% change in the initial requirements. The results show that the growth and change in requirements cause an increase of 38% for the effort, and prolong the delivery time by 60%. The connectedness or relationship can be explained by considering the additional and new requirements that cause a makeover in design. In the process of redesigning and redeveloping the software, additional rework effort is needed, and extra effort is required to detect and remove the newly injected defects. In fact, due to the instability of requirements, the rework percentage has more than doubled, and the productivity has dropped, due to the defect density of the final product has increased. As a result, more effort is needed to make the corrections, and finally the project schedule is delayed considerably.

This ripple effect is a direct consequence of the connectedness between the various components involved in the system, or simply put, it is a chain of cause-and-effect events. In order to model the chaotic reality using a systematic approach, systems thinking works by portraying the system as finite models of reality to study the problems and phenomena that can be observed in the real world. However, the models may have a limited number of components and interrelations, and it results in an inevitable discrepancy between the models and the reality. The models represent assumptions in the relationships of the interacting components, and only if these assumptions hold will the resulting models have predictive values. Therefore, systems thinking as a discipline encourages making key assumptions in models as explicit as possible (Wendorff 2002). In general, the benefits of systems thinking include:

- Ability to deal effectively with problems, and raise our thinking to a higher level at which we see the system as a whole by understanding the complexity through the interactions and connectedness of components

- Ability to study the chain of cause-and-effect events unfolded by tracing the connectedness of the components and to provide an in-depth analysis with insights of the relationships
- Ability to generalize the concept into all fields such as sustaining a business, supply chain management, dynamic behavior of stock markets, and effective management policies and decision making, just to name a few

Nevertheless, systems thinking is just a school of thought, and applying this approach to something useful is another aspect that we can't neglect. Hence, this research is trying to apply systems thinking in system dynamics modeling. The following sections provide an introduction of its applications.

2.3.1 Causal Loop Diagram

Although systems thinking is a powerful concept to gain insight into complex problems, our mind is not powerful enough to visualize the inter-connectedness of the elements without the help of diagrams. Hence, a diagram called causal loop diagram, is used to describe this type of cause-and-effect relationship. Causal loop diagram is the heart of systems thinking. It defines the key relationships and components in the system, and provides a basic understanding of the feedback concepts. Feedback is a continuous flow of information within a system, and it has a property of self-correction. For example, our normal body temperature is maintained at 36.9°C. We have internal mechanisms to control and maintain this setting by shivering when it is too cold, and by sweating when it is too hot. This is accomplished through a feedback system starting with the hypothalamus in the brain that measures the body temperature. Any change in the system will trigger a response, and hence the shivering or sweating, in order to retain equilibrium.

To illustrate the workings of causal loop diagram and feedback concept, the following figure shows the relationships of workforce and effort. The "s" notation shows a change in a causal variable creates a change in variable it is affecting in the same direction, and the "o" notation shows the opposite direction. An arrowed circle

notation is used to portray the feedback loop. The concept of a feedback loop depicts that any variable in a system will be affected by its own action and those variables involved in the loop.

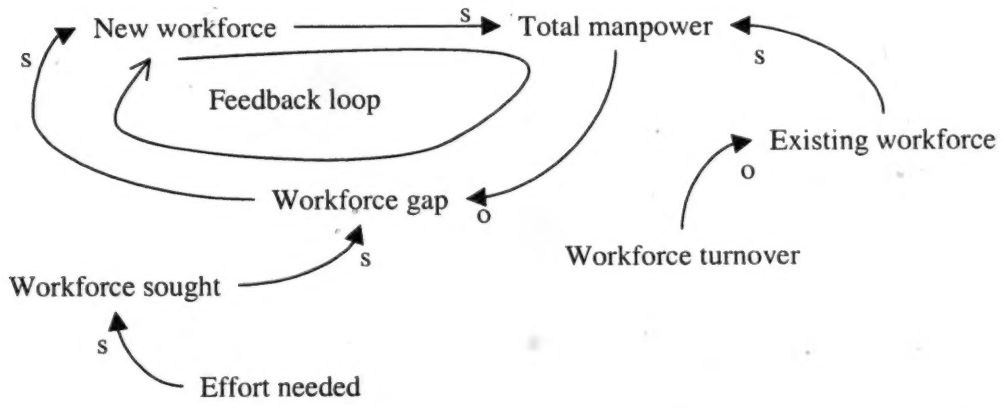


Figure 2.1: Causal loop diagram

When an effort is needed to complete a project, the workforce sought is determined based on the extent of the effort. The workforce gap is then based on the current total in-house manpower and the workforce sought. If there is a lack of manpower, new workforce may be hired to increase the total manpower, which in turn will reduce the workforce gap as a self-correction.

Having the causal loop diagram is insufficient to fully explore the applications of systems thinking. Commonly, the relationships involved in the systems thinking may grow to a stage that is too complex and overwhelming for humans to comprehend the full extent of the models. Hence, computer modeling is used to simulate the models using system dynamics as described in the next section.

2.3.2 System Dynamics

Systems thinking explores the interdependencies among the elements of a system as a whole. Systems thinking looks for patterns and focuses on the feedback loop structure of a system because that structure determines the system's behavior over time. System dynamics is the necessary foundation underlying systems thinking. Systems

dynamics deals with how things change through time and interpreting the system by simulation models that allow one to see how the structure and decision-making policies in a system create its behavior.

As we understand, the multi-project management is dynamic and complex in nature, incorporating various feedback processing and control within an organization. Often, beside the technical factors, the project development also involves people and environments in the feedback loops. Hence, much of the feedback control is indirect, unplanned, or even unconscious. Since the feedback mechanisms are non-deterministic in nature, modeling or simulation must therefore replace the analytical tools of control theory, and system dynamics is considered appropriate for the investigation (Lehman 1998). This approach has been used and well established for over forty years since Forrester showed in his book how a holistic systems thinking approach can throw great light on a host of problems (Forrester 1961). System dynamics looks at exactly the same kind of systems from the same perspective as systems thinking. They both construct the same causal loop diagrams, but system dynamics takes the additional steps of constructing and running a simulation model, and testing alternative policies in the model. Systems dynamics places its emphasis on structure and the processes within that structure to model systems identified in the real world. This methodology assumes a machine representation and uses 'stocks and flows' to illustrate complex interactions within existing systems. As a result, the knowledge of the interrelated technical and social factors coupled in the simulation tools can provide a means for the organization to manage the projects more effectively and predictably.

The purpose of this section is to give an overview of the system dynamics principles. This technique has been proposed and used to model real world socio-technical and other complex processes to support policy making and assessment (Coyle 1996). System dynamics is a tool that can assist the managers to deal with the systematic properties of the project environment.

Levary et al. (Levary 1988) suggested the following objectives in simulation modeling:

- To understand the relationships within a complex system.
- To experiment with the model to assess the impact of actions, options, and environmental factors.
- To test the impact of various assumptions, scenarios, and environmental factors.
- To predict the consequences of actions on a process.
- To examine the sensitivity of a process to internal and external factors.

System dynamics uses mathematical, non-linear differential equations, which translates into a quantitative approach. It was used for the first time in the software development process by Abdel-Hamid and Madnick (Abdel-Hamid 1991). In order to analyze, manage and control the behaviour of a system, the basis of this technique is to understand the cause and effect relationship of the variables that affects the behaviour using causal loop diagrams. To incorporate the diagrams into a system dynamics model, the diagram is transformed into a flow diagram that has additional elements of level, rate and auxiliary variables. The following illustration (Figure 2.2) is converted from the same figure in the causal loop diagram (Figure 2.1). The level is represented by a rectangular, the rate by an hourglass and the auxiliary variable by a circle. The source and the sink are external to the project environment. The hiring rate and turnover rate determine the levels of the workforce, and the rates may be associated with certain time delays, like hiring delay and assimilation delay, which represent lagged relationship in the system where the causal effect is not instantaneous. The levels are used to calculate the auxiliary variables, like the total manpower, workforce sought, workforce gap and effort needed. In return, the auxiliary variable, like workforce gap, affects the hiring rate of the workforce. This interaction results in a feedback loop, as described in the causal loop diagram.

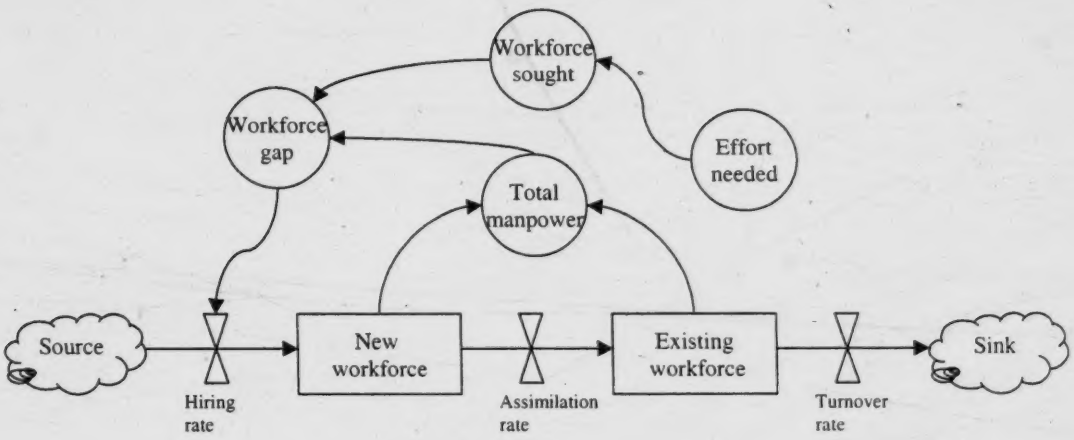


Figure 2.2: Flow Diagram

2.4 SCENARIO MODEL

Systems thinking is rendered useless if it does not equip with the capability to model different situations using different parameters to generate various possible outcomes. During the project development, different scenarios with different events, policies, theories and strategies may be encountered. A project manager can plan for the expected behavior of a project development process but the unexpected scenarios, such as a spike in attrition rate due to better job opportunity elsewhere, may affect the project development in terms of schedule and costs. Hence, the uncertainties must be considered in the development plan as the possible outcomes. By having a Scenario model in the simulation, the manager can test the effects with several combinations of events on the process. It is a model that allows a project manager to define several different scenarios for the project development. Scenarios represent events, policies, procedures, actions, and strategies that cannot be considered part of a development project, but practices imposed or applied to the project and exceptional situations that the manager might encounter during project development. (Barros 2000). These four major categories are described as follow:

- Events: these scenarios include the uncertainty events that may alter the project original behavior and affect the project attributes, like the completion

date, total effort required, productivity and quality. Event scenarios are associated to specific project elements, such as project size, technology, developers' roles, artifacts, and application domain. For example, the requirement volatility can be high in software industry. If an event occurred that changed a significant portion of the requirements, there would be a number of alterations in the project behavior due to this change, such as schedule, productivity, and workload, just to name a few.

- Policies: these scenarios represent the management policies and procedures imposed on a project. Some of these policies are the extensions or outcomes of the organizational mission statements and goals. Like the event scenarios, policy scenarios are also associated to specific project elements. Examples of policy scenarios are the delays to hire new employees, staff turnover, availability of resources in an organization, and reward schemes for the employees. The policies represent the tactical behavior patterns that can affect the project environment. For instance, if the company's goal is to continuously seek improvements in its workers at all costs, then substantial amount of investment needs to be made in training the workers. As a result, in a financial good time, the organization may benefit from this policy by having well-trained and highly productive workers. However, when the company suffers financial loss, the expensive training may further cripple the existing weak financial status.
- Theories: these scenarios involve the proven or the hypothetical management theories imposed on a project that may impact its behavior patterns. A project manager may study the effectiveness of a theory, using scenario modeling to observe the patterns and behavior. For instance, Abdel-Hamid's error propagation theory (Abdel-Hamid 1991) may be represented as a theory scenario. The other theory scenario is Brooks' law, which will be further explained in Section 3.3.
- Strategies: these scenarios represent the decisions and action plans the manager may execute during the project development. These strategies may have short-term and long-term effects on the project lifecycles. Strategies

represent manager's specific decisions that are highly coupled to the elements that compose a specific software project. For example, allocating more resources to an activity, multi-tasking for the developer, creating internal milestones before external deadlines, and imposing false schedule pressure on the developers.

The scenario model defines the alternative routes that the project may encounter due to the unexpected events. By capturing different scenarios in a model, the managers are able to build a reusable knowledge base for the project management. This information can be reused to other projects associated with these elements (Barros 2000).

Scenario models are abstract in nature, and thus cannot be directly simulated. They must be integrated into some project models, like system dynamics models, before simulation analysis can begin. This integration occurs through an intermediate interface that channels information or variables to appropriate project models that the scenario models can act upon. In this research, the scenario modeling is extended to the system dynamics models, based on the 4 major categories provided by Barros, using a controller as the intermediate interface. Barros only provided general explanations of these categories, and our work is to further explore several specific project features or elements that are tied to the above-mentioned scenarios, using the system dynamics models as the foundation. Several project elements have been identified and explained in Section 3.3. In this section, variations of these elements may generate different scenarios and their impacts on the system dynamics models are then determined.

3.0 MULTI-PROJECT MANAGEMENT MODELS

In this research, there are 3 types of methods being employed to create a coherent multi-project model. Those methods are multi-project network, system dynamics models, and scenario modeling.

3.1 MULTI-PROJECT NETWORK

A multi-project network is needed to build a base for a network of concurrent projects, so that it can represent sequence of activities, allocate resources and model the effect of activity schedule on projects sharing the same resources. The methodology used to achieve this is one of the newest methods in the project management paradigm, called Critical Chain Project Management (CCPM), introduced by Goldratt (Goldratt 1997). Unlike PERT/CPM (Program Evaluation and Review Technique/Critical Path Method) that assumes infinite resource capacity, CCPM involves the resource allocation and leveling across multi-projects and buffer placement within the critical chain. It uses a common method to identify the critical chain in a multi-project network, and recognizes the interdependencies of various projects in the resource requirements, and accommodates the nature of individual projects in resource allocation and project buffering. Hence, this methodology complements and suits the goals of this research and its fundamental concept is applied here as described in the following subsections.

There are basically 4 steps to construct CCPM network:

- The development of a project network using PERT
- The resource leveling using modified Wiest and Levy heuristics
- The development of the critical chain
- The proper placement of the buffers

To illustrate the multi-project network, an example related to software engineering is used for this purpose. Two simple projects, Project I and Project II, are involved in an organization with two different styles of software lifecycles. Project I is an

incremental model type project, while Project II uses the traditional waterfall model approach. Project I has two increments in the development and is relatively bigger in size and higher in priority than Project II. Project I planned start date is 32 weeks earlier than Project II.

3.1.1 Step 1 - Development of Project Network

PERT chart can be used as the foundation of the project planning to show the technological sequencing of activities (Walker 2000). All individual projects are constructed under a PERT chart style network. Each project activity is listed out with the consideration of the sequencing order. For Project I, the requirement activity is conducted first, followed by two increments of the design and coding. Upon completion, the integration phase is commenced and followed by the testing phase. For Project II, since the project is relatively small, the sequence of the activities is straight forward, i.e., requirement, design, coding and testing in a respective order. The properties of these two projects are summarized in Table 3.1:

Project I	Project II
Incremental model	Traditional waterfall model
High priority	Low priority
Big project size	Medium project size
Start date at 0 week	Start date at 32nd weeks
Increments = 2	Increments = 0

Table 3.1: Project I and II Properties

Each activity has an estimated duration provided by the individual project manager. It is common in software engineering to have variations of activity and project duration during the project development. The underlying reasons will be considered in the system dynamics model. These variations of activity and project duration may affect the project schedule and resource allocation. In the mean time, during the project planning, the project manager can also reduce the impact of this effect on the schedule by inserting the buffer in Step 4, which will be discussed later. Hence, at this

point of developing the project network, it is sufficient to provide each activity with an estimated duration. (Note: The procedure to determine the best duration estimate is beyond the scope of this research. However, it is common to use triangular probability distributions to determine the expected completion date.)

After determining the activities, the duration and the sequence order, the critical path of each project is established using the following steps: (Walker 2000)

- Make a forward pass through each project to calculate the early start (ES) and early finish (EF) for each activity starting at the first activity in the project.
- Using the EF for the last activity, make a backward pass through each project to calculate the late finish (LF) and late start (LS) for each activity starting at the last activity in the project.
- Slack, S, is calculated by subtracting EF from LF, or subtracting ES from LS.
- Critical path is the path with no associated slack.

Let the planned start date, Ps, for the first project to start in the multi-project network equal to zero. In the example, Ps I =0. Then, for the remaining project, let the planned start date equal to the number of periods from the point of reference of the first project. For Project II, Ps II is 32 since it is scheduled to start 32 weeks later than Project I. Then, recalculate those projects with Ps greater than zero, by adding the Ps value into the ES, EF, LS and LF parameters. This adjusts the multi-project network to reflect the staggered start dates.

The following diagram (Figure 3.1) describes the first step in creating a multi-project network.

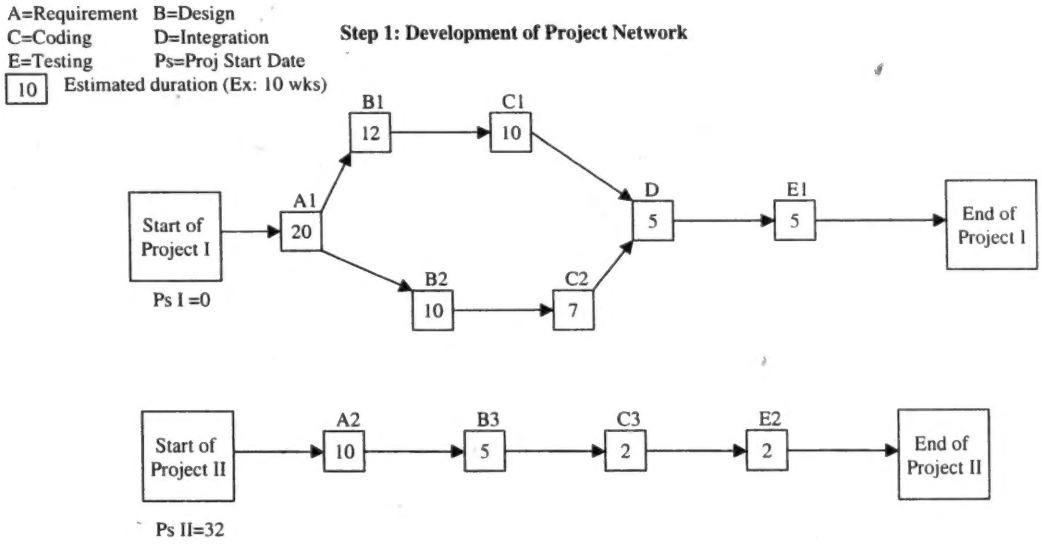


Figure 3.1: Development of Project Network

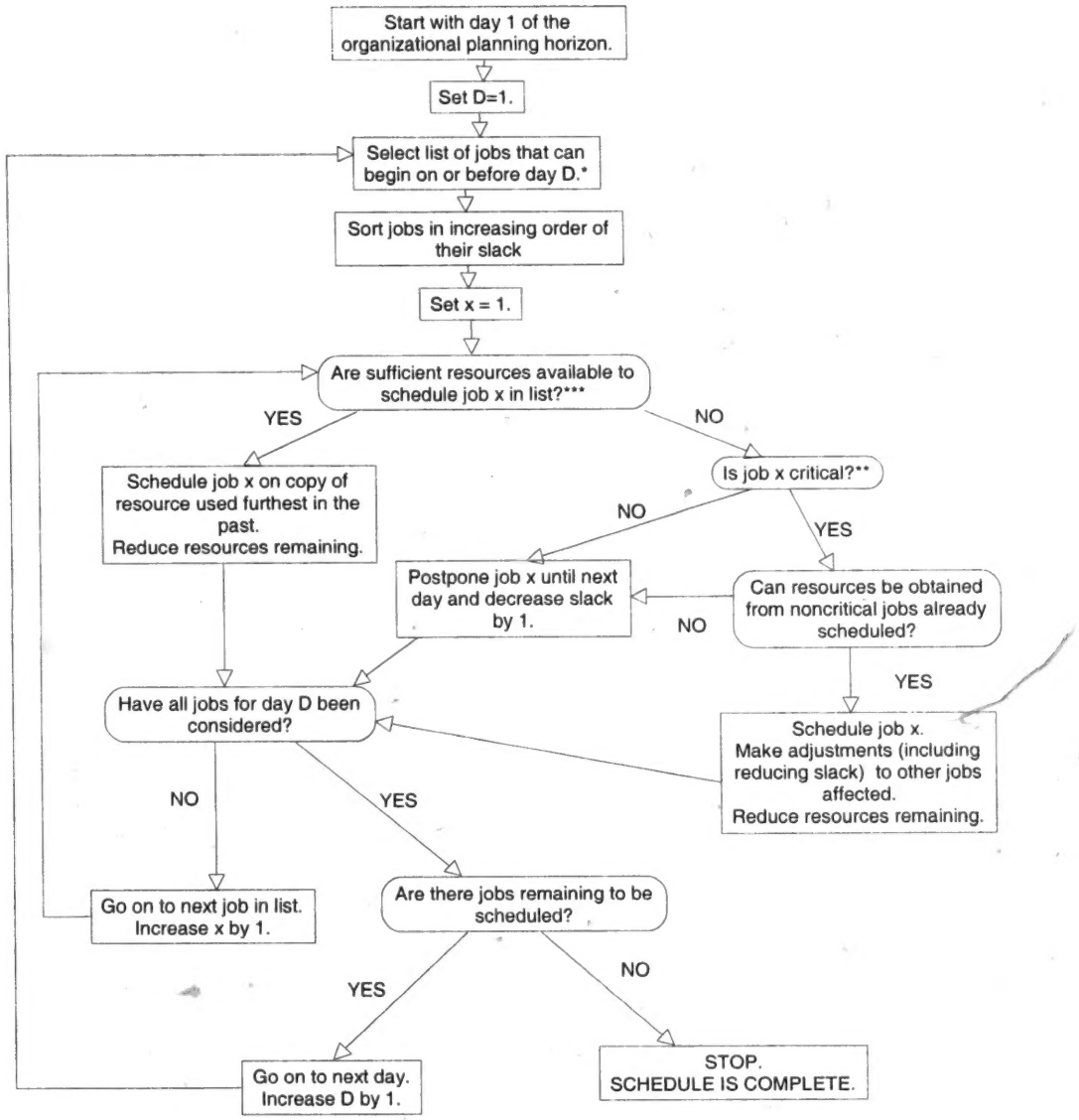
Before starting Step 2, we create a dummy activity called "Start of planning horizon" prior to the earliest start date for all projects and create a dummy activity named "End of planning horizon" after the latest early finish date for all projects (Walker 2001).

3.1.2 Step 2 - Resource Leveling

The first step, however, does not take into the consideration of the finite capacity of the resources, and thus, the effects of resource contention are hidden from the project network. For example, B1 and B2 activities use the same resource, and obviously, both activities cannot be carried out at the same time assuming the resource pool has only enough to accommodate one activity at a time.

Wiest and Levy heuristics (Wiest and Levy 1977, Walker 2000) provides a method to allocate the resources. Figure 3.2 provides an overview of the heuristics as deployed with a CCPM network. Whenever a type of resource is required on the same day across multiple activities, the heuristics assigns the resource to the activity with the least slack time. Slack is calculated by finding the difference between the earliest possible start date and the latest possible start date of an activity. If an activity is critical and short of resources, the resources are obtained from the non-critical jobs

already scheduled. If the resources from other scheduled tasks are not available, the critical activity has to be postponed. Although the heuristics does not lead to the optimum solution, the approach provides a feasible solution. Basically, this method connects the activities across different projects that use the same resource to indicate the resource precedence relationship. Figure 3.3 applies this approach to the sample problem. For example, the resources are first assigned to the activity E2 since the earliest start date for E2 is earlier as compared to the earliest start date for activity E1. In other words, the demand for the resources comes first for the activity E2, and only upon completion of the activity E2, the resources are shifted to the activity E1. The ordering is identified by the resource sequence arrows.


Start with day 1 of the organizational planning horizon.
Set D=1.
Select list of jobs that can begin on or before day D.*
Sort jobs in increasing order of their slack
Set x = 1.
Are sufficient resources available to schedule job x in list?***
YES
NO
Schedule job x on copy of resource used furthest in the past.
Reduce resources remaining.
Is job x critical?**
NO
YES
Postpone job x until next day and decrease slack by 1.
Can resources be obtained from noncritical jobs already scheduled?
NO
YES
Have all jobs for day D been considered?
Schedule job x.
Make adjustments (including reducing slack) to other jobs affected.
Reduce resources remaining.
NO
YES
Go on to next job in list.
Increase x by 1.
Are there jobs remaining to be scheduled?
YES
NO
Go on to next day.
Increase D by 1.
STOP.
SCHEDULE IS COMPLETE.

* Day D is the day under consideration.
** A job is critical if it has no PERT/CPM slack with respect to its individual project.
*** Job x is the job under consideration.

Figure 3.2: Modified Wiest and Levy Resource Leveling Heuristics

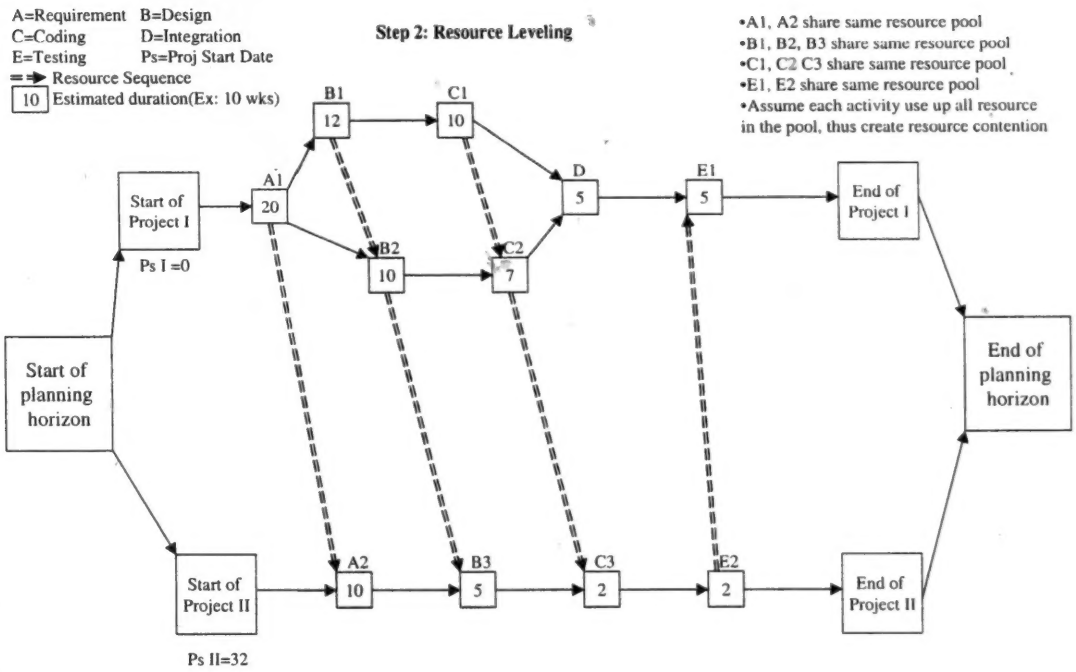


Figure 3.3: Resource Leveling

3.1.3 Step 3 - Development of the Critical Chain

The critical chain can be determined considering not only the activity sequencing but also the resource contention (indicated by the resource sequence arrows in Figure 3.3) (Walker 2000). Hereinafter, the term "critical path" will be used to denote a sequence of critical activities in a multi-project network that recognizes only technological precedents, and the term "critical chain" will be used to denote a sequence of critical activities that recognize both technological precedents as well as the use of common resources. The slack time is then calculated using the same method as in Step 1. Critical chain is the longest sequence of critical activities in a network that considers both the technological sequencing of activities and the simultaneous demand of common resources (with the least total slack). Any delay on the critical chain will delay the project duration that may have ripple effect to the other projects that share the same resources. The critical chain is determined using the Critical Chain Multiple Project Environment Completion method (CCMPC) (Walker 2001). It uses the similar mechanism to find a critical path in a PERT network by finding the least slack

path, and in addition, considers the resource sequence for all related projects. The results of using the CCMPC method show that the Project I's critical chain is A1-B1-C1-C2-D-E1, and for Project II is A1-B1-B2-C2-C3-E2. It is obvious from the example that every activity in Project I is critical and may affect Project II if there is any delay, while Project II is only critical in the last two activities.

3.1.4 Step 4 - Placement of Buffers

A schedule can be designed to protect the project completion date by "placing slack" amongst the tasks, as buffers. As stated before, the project manager can place the buffers at strategic points in the project to reduce the impact of the variation effect on the schedule. Buffers are used to ensure the success of a critical chain and the overall project to prevent a non-critical activity delaying the start of an activity on a critical chain. They provide early warnings regarding the progress of a project, and then further planning and actions are taken depending on how many buffers have been consumed by the project development.

A convergence point in a project network is first determined at those critical points where two or more activities must be completed prior to the start of a subsequent activity in a critical chain. There is one such critical point in the example, where activities B3 and C2 converge on C3. Since both activities must be completed before the subsequent activity in a critical chain may begin, a delay in any preceding activities will delay the subsequent activity. As B3 is not on the critical chain, we must prevent B3 from delaying the start of C3 by inserting a "safety buffer", so that a delay in the completion of B3 does not impact the critical chain or the subsequent critical activity C3.

There are 3 types of buffers (Walker 2001). Project completion buffer, PCB, is placed between the last activity and the end node of the project. This buffer is to reduce the variation in the overall project network to protect the expected project completion date. Convergence buffer, CB, is placed where a non-critical chain intersects the critical chain of a project (as described in the above paragraph). Resource contention

buffer, RCB, is a type of convergence buffer and is placed between the use of a common resource that is needed on two different chains, one of which is the critical chain. For example, if B3 and C3 use the same resource, then the buffer placed in between them is RCB. The first two types of buffers can be found in Figure 3.4.

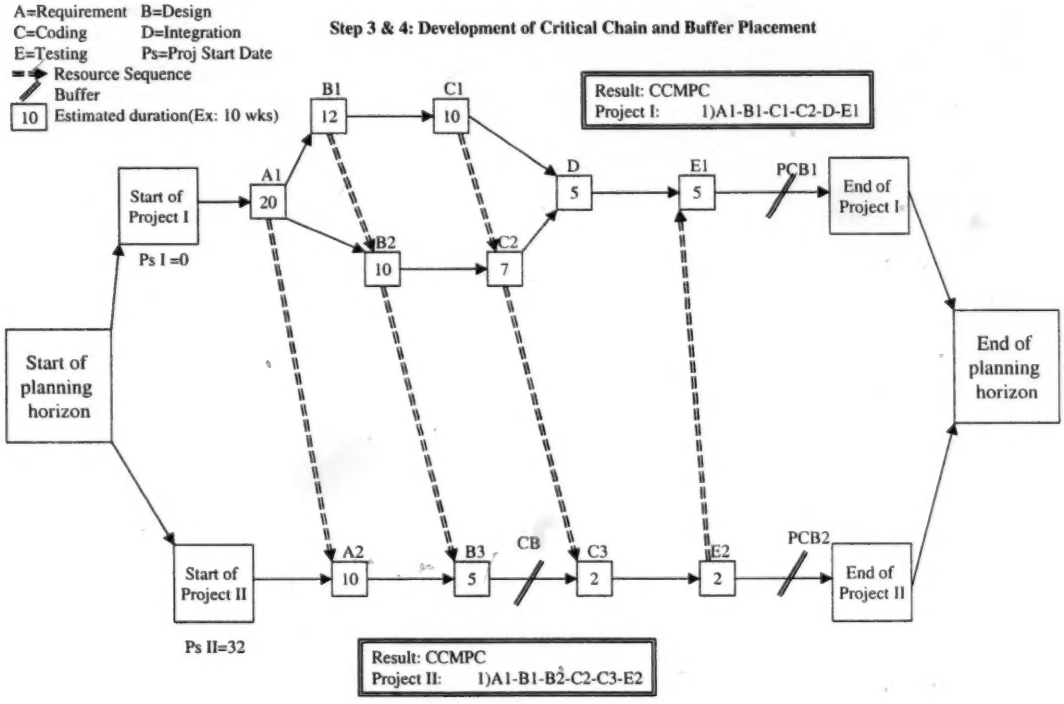


Figure 3.4: Development of critical chain and buffer placement

The same steps are undertaken for other concurrent projects. Upon completion of the network development, the multi-project network provides a common ground for system dynamics models to act upon and spawns a link between the models to initiate cause-and-effect relationships. The details of the workings of the link are explained later in Section 4.

3.2 System Dynamics Models

Multi-project environment in software engineering may play an important role in the management of human resource and software productivity. The relationship that exists among the environment, software development and human issues is yet unclear and can be best described using system dynamics. For instance, allocating people to projects in multi-project environments is difficult, and the more projects that are involved, the more important are the allocation process (Hendriks 1999). However, having a single system dynamics model is insufficient to portray an entire complex system. Different inter-related models are needed to link causal loop diagrams into chains of cause-and-effect events. The models represent the basic sub-systems that provide abstraction to the intricacy of a holistic system.

The models may help us to understand how simultaneous projects influence the effort needed in software development, and how this effort affects the planning and control of the projects. The planning stage determines the number of workforce required in the human resource. Different human resource policies result in various degrees of hiring and attrition of staff, which in return may have an impact on the overall productivity of the organization. Positive and significant effects on labor productivity are found for organizations that utilize more sophisticated human resource planning strategies (Koch 1996). Hence, the chain of relationships goes on, and without defining a definite boundary, the system may grow too big with a tendency to describe the problem in a colossal view. This could include literally everything, and so this is unhelpful, as the relationship gets too complicated to handle. By drawing a boundary, we can better understand our system of interest, without having to worry about too many unrelated details. Thus, in this study, with a confined boundary, the system dynamics models that need to be developed only include:

- Workload and exhaustion model: To study the increasing or decreasing workload on the staff's exhaustion level
- Human resource model: To model different human resource policies, including hiring and firing in software engineering organization

- Effort model: To understand the effect of various project related factors on the effort needed to develop software
- Productivity model: To define an overall software development productivity based on staff's ratio, ability, exhaustion level, communication overhead and ease of development
- Control model: To assess the progress and determine the remaining effort perceived still needed to complete the tasks
- Planning model: To provide initial planning at the start of the project/activity and readjust the schedule and workforce as necessary throughout the lifecycle

In a multi-project environment, there are variables and factors that are shared among the projects running concurrently. For instance, in the human resource model, the resource pool is a mutual location to look for available workforce, and factors like the policies of hiring and firing often affect the entire organization, and thus influence all existing projects. Similarly, the workload and exhaustion model exerts the same mutual and accumulated effect on the development staff, i.e., when more projects are given to the limited amount of staffs, the increased pressure and workload will impact the well being of the employees and eventually may exhaust them within a certain time period. These two models are thus called "Shared Models", as the components in these subsystems have mutual effects on all projects.

Conversely, there are variables that are only unique to their individual projects. For example, each project is different in size, complexity, requirement volatility, and effort needed to complete it. Hence, the planning and control of the project is unlike any other project. So, the models associated with the individual projects are not shared as the two models described above. They are unique and separated from other projects, and they are called "Individual Project Models". The following diagram illustrates the concept. In the subsequent explanation and illustration, only two concurrent projects are used to explain the subsystem models.

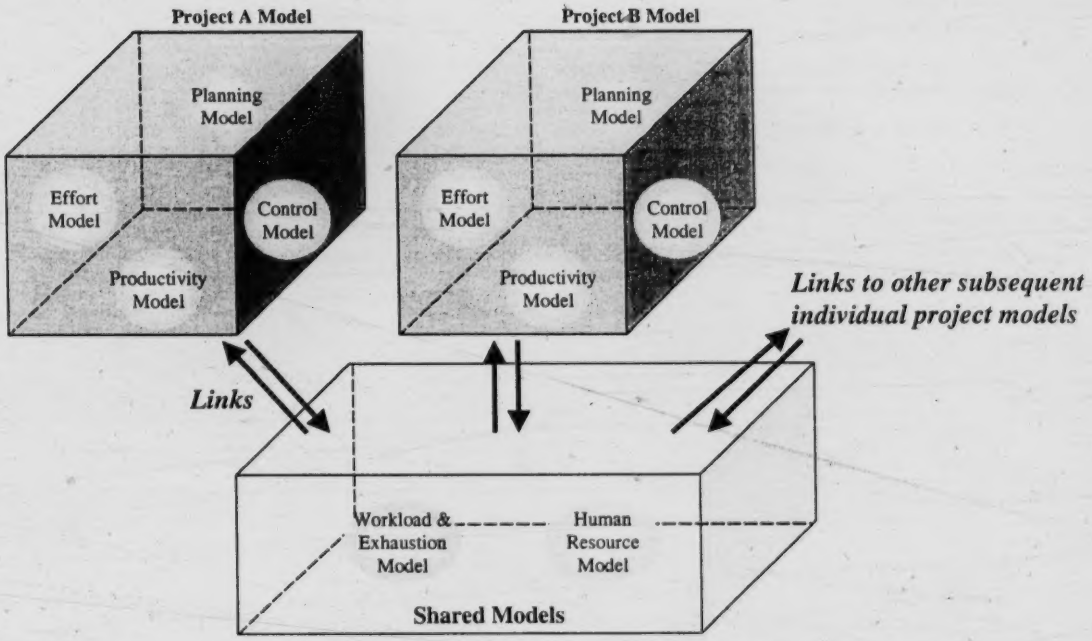


Figure 3.5: A Simplified View of Multi-project Subsystem Models

3.2.1 Shared Models

There are two models that are shared, i.e., Workload and Exhaustion model and Human Resource model. The causal loop diagrams that are used to explain the models incorporate the two concurrent projects, Project A and Project B. For those variables, that are unique to individual projects, have a subscript indicating either "A" or "B" to represent the two projects respectively. For example, "$Tasks_A$" refers to the number of tasks in Project A. However, if the variable "$Tasks_{A,B}$" is mentioned in the article with two subscripts, then it means the description is referring to all concurrent projects, i.e., "$Tasks_A$" and "$Tasks_B$". If there are no subscripts indicated in the variables, then they represent shared components in the system. Finally, if a variable is italicized in the diagram like "*Total workforce*", it means the variable is external to the currently mentioned model, and it represents a link to another model.

3.2.1.1 Workload and Exhaustion Model

This model is shared with other individual models as workload and exhaustion levels are simultaneously affected by various projects. The level of exhaustion is monitored throughout the development of projects, and is influenced by workload. This model also reveals the impact of workload and exhaustion on the quitting of personnel, and studies the factors that cause the build-up of exhaustion. Figure 3.6 shows the causal-loop diagram of the model, while the corresponding flow diagram is illustrated in Figure 3.7.

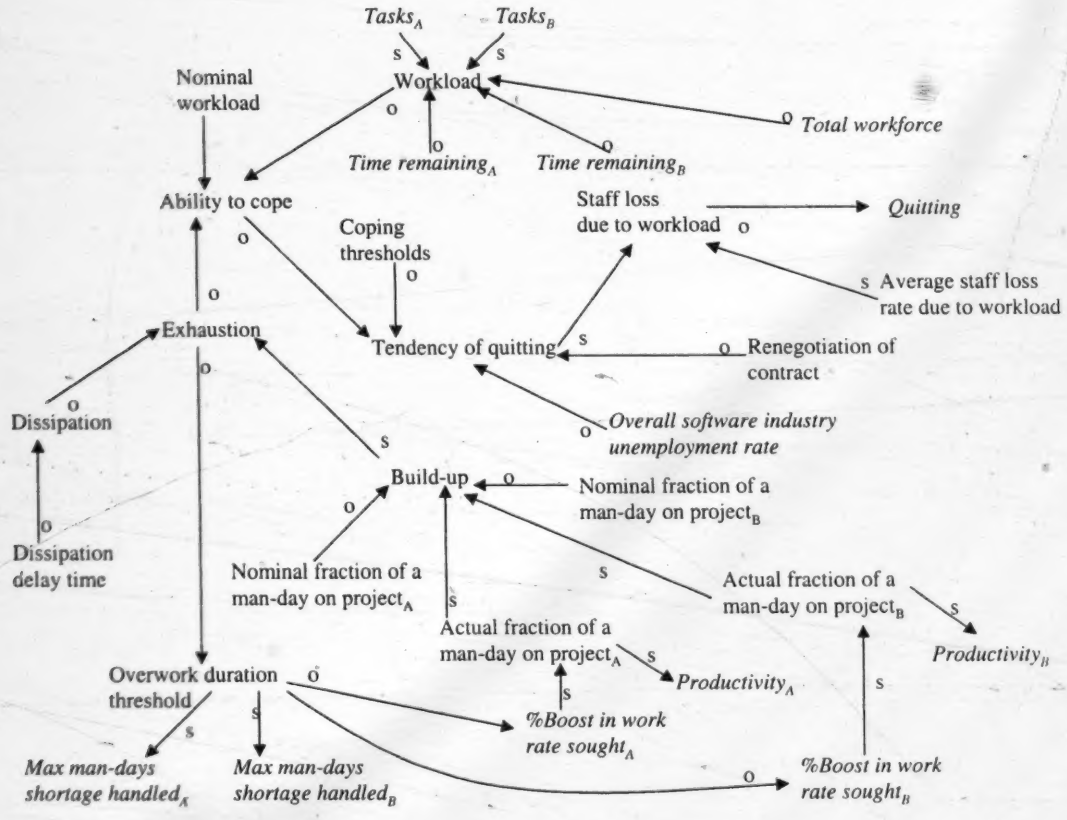


Figure 3.6: Workload and Exhaustion Model in causal-loop diagram

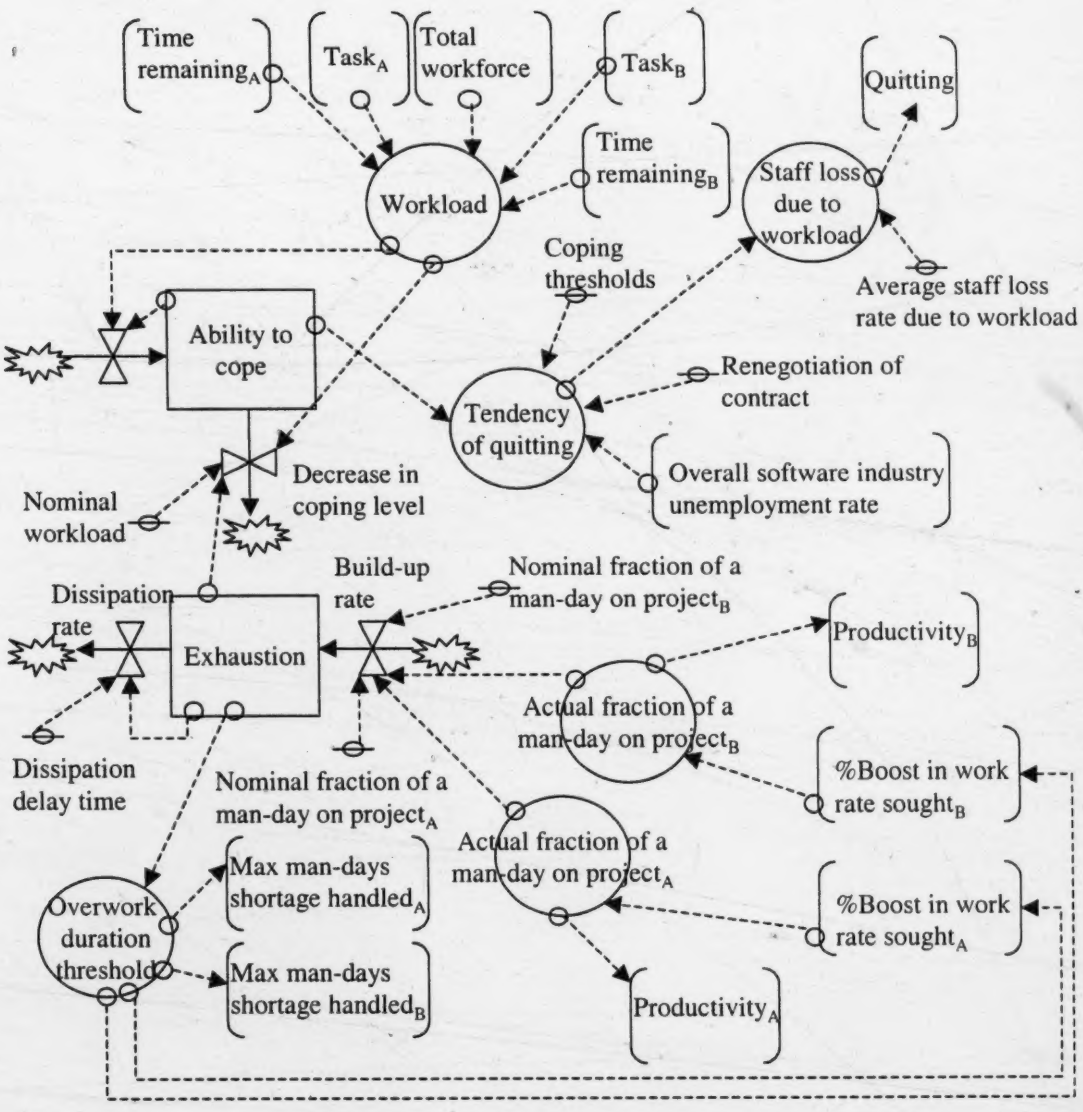


Figure 3.7: Workload and Exhaustion Model in flow diagram

Workload per capita is a measure that is used by managers to assign tasks, and is also perceived by workers as a responsibility in an organization. As the number of "Tasks$_{A,B}$" increases, the effect on the "Workload" is mutual. Workload is determined by the total number of remaining tasks of all projects being distributed over the "Total workforce" in the organization that needs to be completed in a certain amount of time allocated, defined as:

$$\text{Workload} = \frac{\sum_{i=1}^{n} \frac{Task\ remaining_i}{Time\ remaining_i}}{Total\ workforce}$$

For example, if the total number of tasks in all projects is 200 tasks, having a workforce of 20 full-time employees, and the projects are to be completed in the remaining 10 working days, then the workload per person is 1 task per working day. However if the two projects have different completion dates but same starting date, for example, Project A is to be finished in the remaining 10 working days with 100 tasks remained and 20 workers, and Project B has 20 working days remained with 100 tasks and the same 20 workers, then, if both projects are still on-going, today's workload will be 0.75 task per working day per person, i.e., 0.5 and 0.25 tasks per working day per person for Project A and Project B respectively.

Throughout the project development, new projects may be introduced and old projects may be in the completion phase, and consequently, the workload may fluctuate from time to time as the human resources are loaded and unloaded with ongoing tasks. This inevitably affects the workers' ability to cope with their jobs. Generally, the average worker's ability to cope with responsibility is inversely related to the amount of "Workload" (Sherwood 2002). Although there are exceptions where "busy" workers are constantly loaded with more work due to their capability, the relationship described in Figures 3.6 and 3.7 shows the common response in average workers where the ability to cope is an indirect measure of the well being of the workers that is being affected by the workload. According to a survey in News Ltd Metropolitan Newspapers (Warren 2003), 46% of the workers are reporting that the workload is damaging their health and well being, and reduces their ability to cope with their work. Hence, it is assumed that there is a "Nominal workload" that the workers feel comfortable to work in. Any actual workload higher than the nominal may result in a decrease in their ability to cope with their work. We formulate the following example to illustrate this effect in Figure 3.8:

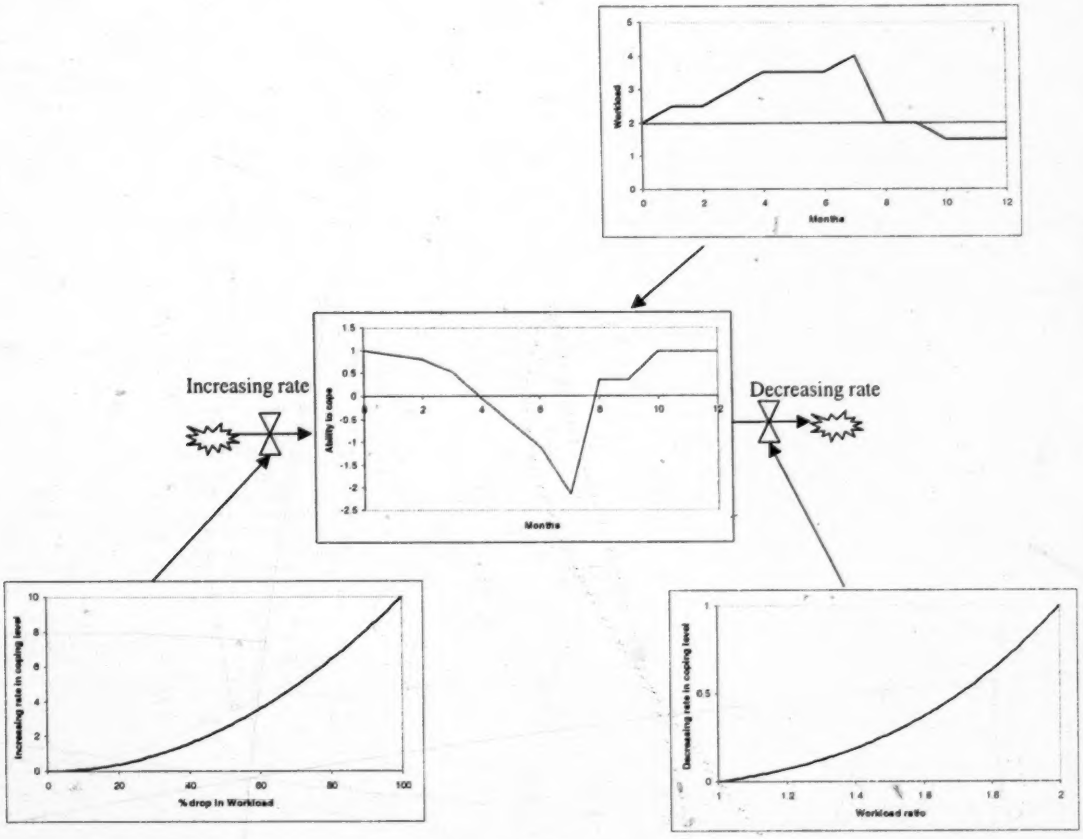


Figure 3.8: Ability to cope

"Ability to cope" is simply a level whose value reflects the level of ability to handle the job, and is assumed that when the level is dropped below 1, the workers may have tendency to quit, and the lower the level, the higher the tendency. A "Nominal workload" is defined as the workload of the average workers can handle without decreasing their ability to cope. In this instance, the nominal workload is set at 2 tasks per person per working day, and at the start of the project development, the ability to cope is set at 1. Workload ratio is determined by dividing the current workload by the nominal workload. Hence, when the ratio is greater than 1, it means that the workers are being stressed to work more than the nominal level. When the ratio is slightly more than 1, the effect of the stress may not be significant to be felt by the workers, but as the ratio goes higher, the workers may feel the effect and result in a bigger drop in the coping level, as shown in the graph at the lower right corner of Figure 3.8. On

the other hand, the ability to cope level may rise when there is a reduction in the workload. The percent drop of workload is defined as follows:

$$\%\ \text{drop in workload} = \frac{\textit{previous workload} - \textit{current workload}}{\textit{previous workload}}\%$$

Similarly, when the percent drop is small, the workers may not notice the reduction, and thus the level to cope may not rise as fast as when the percent drop is higher. This is illustrated in the graph at the lower left corner of Figure 3.8.

Initially in the first two months, the workload is set at 2.5 as shown in the top right corner of the figure. As the current workload is close to the nominal workload, the decreasing rate in coping level is relatively small in the first two months. Starting in the 3rd month, the level drops further as the effect has been felt when the current workload is 50% more than the nominal workload. Continuing the same workload into the next few months, results in a further drop of coping level, as shown from the 4th month to the 6th month. By the end of the 6th month, the workload has increased by 2 times from the nominal, and the workers may experience a sharp decline in coping with the responsibility. After the 7th month, the workload has started to reduce. At first, on the 8th month, as the percent drop in workload is relatively large, the coping level recovers quickly. The next month shows no change in workload, and as a result, the coping level maintains at its previous level. On the 10th month, the workload is still less than the nominal, and the ability to cope improves further. Subsequently in the last two months, the coping level maintains as it is when the workload is kept constant.

The "Tendency of quitting" can be determined by the following three factors, i.e., the "Ability to cope", "Overall software industry unemployment rate", and "Renegotiation of contract". The effects of each factor on the quitting tendency are tabulated in the following table.

Tendency of quitting	Ability to cope	Unemployment rate	Renegotiation of contract
None	>= 1.0	High	Complete success in negotiation
May be	>= -1.0, <1.0	Medium	Partial success
Likely	< -1.0	Low	Complete failure

Table 3.2: Tendency of quitting

When the workload is reasonable, the "Ability to cope" may be high, and so the "Tendency of quitting" may be unlikely or none. When the "Coping thresholds" are given as above, the workers may not quit if the coping level is higher or equal to 1. When the level is between –1.0 and 1.0, they may have some tendency to quit, and any level less than –1.0 may result in the likelihood of quitting. The threshold values are subjected to vary in organization, environment, and industry. This tendency is also influenced by the "Overall software industry unemployment rate". The staff may not resign, despite the increased workload, if the unemployment rate is high, and may continue working in the company. The characteristics of the labor market show that a high unemployment rate in the region reduces the probability of quitting (Ruiz 2002). Conversely, if the industry has a high demand of the workers, they may resign and result in "Staff loss due to workload". To reduce this effect, the organization may opt to renegotiate the employees' contract, by increasing the benefits, salary or promoting the workers, with the purpose of retaining them. Normally, the workers initiate the wage revision or the "Renegotiation of contract", and if the company refuses the wage increase requested, the workers will quit the job (Ruiz 2002).

"Staff loss due to workload" is determined by finding the average staff loss rates in three different situations based on historical data, i.e., the rate when the organization losses a lot of staff, the rate when the loss is normal, and the rate when the loss is insignificant. For example, when the tendency to quit is very likely, we would choose the rate when the organization losses a significant amount of staff, and vice versa.

If the staff insists on leaving the company, the "Experienced Staff" is reduced by the respective amount of people quitting. That is, we are assuming no turnover among the new staff since it is unlikely for a new recruit to quit within a short assimilation period (Abdel-Hamid 1991). Further explanation in this staffing policy will be given in the Human Resource model.

Besides workload, "Exhaustion" level due to overwork also may reduce this ability. The higher the level of exhaustion, the higher the decreasing rate in coping level. When people get tired, their ability to handle jobs will drop. As it is reduced further, the workers may have a tendency to quit. Another survey (Gewirtz 2000) indicates that stress in IT industry is causing 41% of the respondents to consider leaving their jobs. And, if the ability to cope drops further, the tendency to quit may rise to a level that the workers are very likely resigning due to overwork.

In return, the lesser the staff ("Total Workforce") to work on the projects, the higher the "Workload" per capita as the resigned staff's responsibility is distributed to other remaining workforce, i.e., the number of "$Tasks_{A,B}$" remains constant, but the "Total workforce" has decreased. Hence, this workforce cycle is a reinforcing or positive feedback loop. If the workload is increased continuously, the cycle is vicious in nature, as higher staff loss will be experienced along with higher workload. In turn, the ability to cope declines further and may result in more loss of staff (Sherwood 2002).

"Exhaustion" is a level whose value reflects the level of exhaustion of the work force due to overwork (Abdel-Hamid 1991). The rate at which the "Build-up" of the level increases is a function of some measures of overwork. The overwork depends on how much slack time (e.g. coffee breaks, social communications, personal business) is deprived from their work (Abdel-Hamid 1991). For example, in a normal 8-hr workday, the workers on average spend 70% of their time in project development, and 30% is considered slack time. So, the "Nominal fraction of a man-day on $project_{A,B}$" is 0.7. If the environment remains stable, then the "Actual fraction of a

man-day on project$_{A,B}$" is also 0.7. However, if the project is falling behind schedule, and there is a need to increase work rate by a certain percentage, as in "%Boost in work rate sought$_{A,B}$", then the actual fraction will be greater than 0.7. For example, a 25% boost in work rate would increase the work hours from 5.6hr of work per man-day to 7.0 hr, and improves the productivity by an equal percentage. In other words, the overwork is accomplished by reducing the slack time (Barros 2000), and spending more time in project development. As a result, the "Productivity$_{A,B}$" improves as less slack time is spent on the workday, but the workers suffer higher "Build-up" of exhaustion. Hence, the "Build-up rate" is a function of the nominal and actual fraction of a man-day on project. The following diagram shows the rate of increase in exhaustion level (Abdel-Hamid 1991):

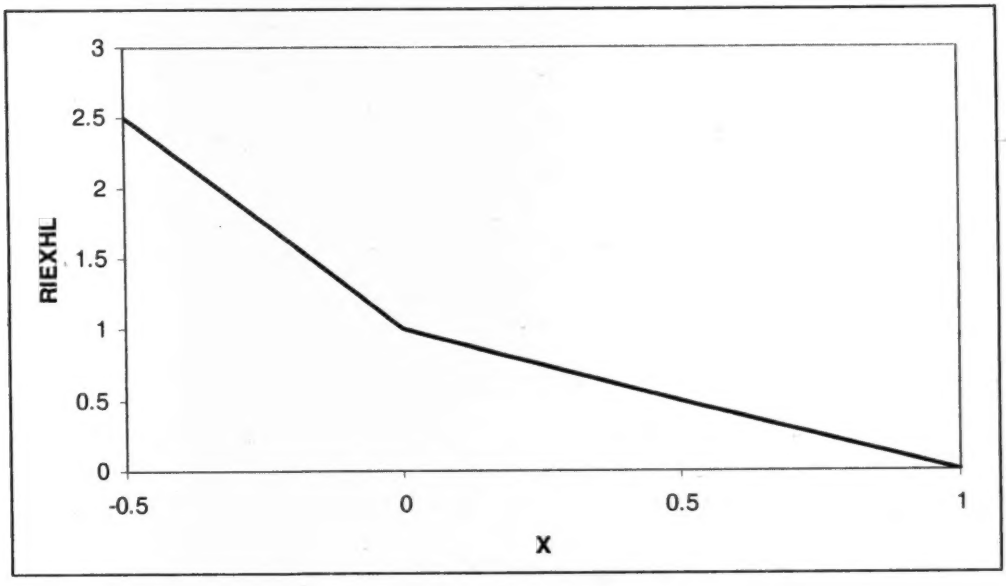


Figure 3.9: Rate of increase in exhaustion level

Where,

RIEXHL = rate of increase in exhaustion level

$$X = \frac{1 - AFMDPJ}{1 - NFMDPJ}$$

AFMDPJ = actual fraction of a man-day on project

NFMDPJ = nominal fraction of a man-day on project

When the X value is between 0 and 1, the workers are compressing their slack time to increase their work rate, and reducing their tolerance for continued hard work. However, when it goes into negative value, the workers are not only compressing their slack time, but also working overtime, and that is why the line in the graph increases at a faster rate.

If the situation persisted, would the workers be willing to work overtime indefinitely? According to Abdel-Hamid, the answer is "no". There is a threshold beyond which the workers are willing to work at an above average rate. This threshold is called "Overwork duration threshold", usually measured in weeks. For example, if the nominal threshold is set at 5 weeks, then the workers are willing to work overtime continuously in that duration. During that time period, the threshold is dropped from a start value of 5 weeks or 40 days. Assume the nominal value for the "Overwork duration threshold" is set at 40 days at a rate of 8hr per man-day. If we maintain the "Actual fraction of a man-day on project" to be 1 for the next 40 days, by the end of the period, the exhaustion level would have increased by 40 units. On passing that duration, the threshold drops to zero and the workers are unwilling to work overtime and are commonly assumed to return to a normal work rate (Barros 2000). Hence, when the workers have achieved their maximum tolerable exhaustion, the "Max man-days shortage handled" is zero, that is the workforce is unwilling to handle any further man-day shortages through overwork.

To recharge the workforce, the "Exhaustion" level needs to be depleted. The "Dissipation rate" is determined by the "Dissipation delay time", and the rate is modeled as a first order exponential delay. During the de-exhausting period, the workers are unwilling to work overtime (Artzer 1982). However, once the period is over, the work force should again be willing to increase work rate if the need arises.

3.2.1.2 Human Resource Model

This section discusses about the personnel resources in the development team, for example, developers, testers, analysts, etc, that are involved in the production of

software. These resources are crucial in an organization as they represent the core of development. The model describes the following relationships:

- How does the workforce transfer affect the entire workforce?
- How do the profits and the cost of human resource affect the hiring, firing and training policies?
- How do these policies affect the total workforce level?

In Figure 3.10, the "Total workforce" level is affected by two key human resource policies in the organization, i.e., "Hiring" and "Firing" (Sharpe 1999). The workforce is assumed to consist of two workforce levels, "New staff" and "Experienced staff". Both new and experienced staff may be hired or fired in these policies. By hiring more employees, the total workforce will increase, and vice versa. The hiring process is triggered when the "Target workforce$_{A,B}$" is higher than the "Total workforce" available, and hence, resulting in a higher "Workforce variance". The target workforce is determined in the Planning model and will consider the willingness to change the workforce level based on the stability of schedule and workforce. Thus, when the variance exists, the organization may choose to hire new staff or experienced staff or both based on individual needs. The equivalent flow diagram figure in Figure 3.11 shows the direct influence of workforce variation on the hiring policies. More detailed explanations about the policies will be explained later in this section.

Figure 3.10: Human Resource Model (General) in causal-loop diagram

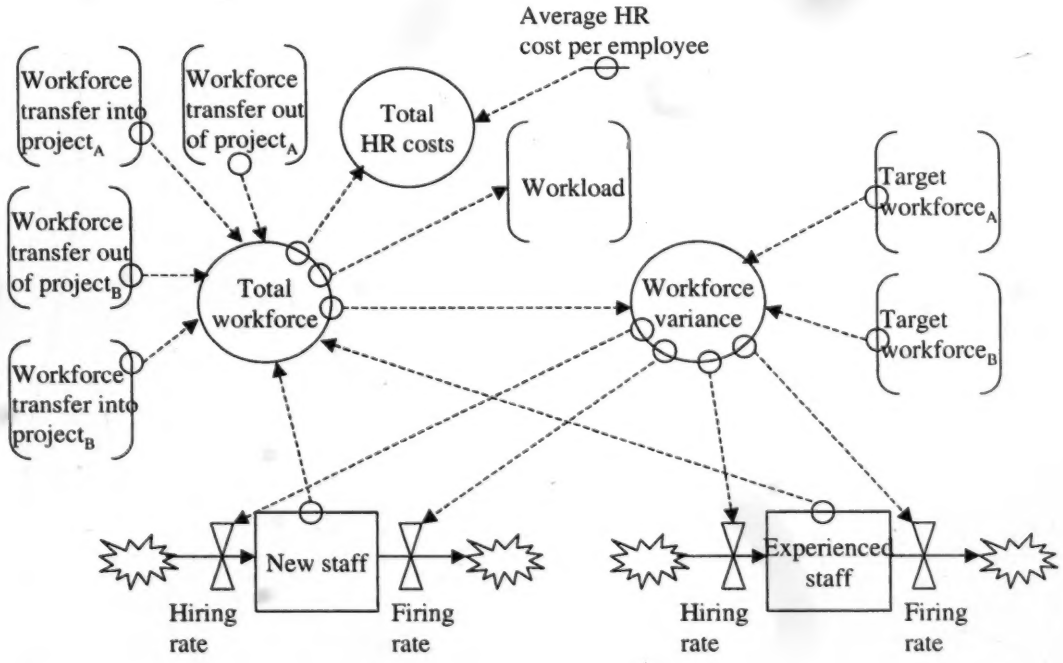


Figure 3.11: Human Resource Model (General) in flow diagram

The same is true for the firing policy. When less people are needed in the organization, the effect on the "Workforce variance" may cause more people to be fired during a downturn in order to reduce the "Total workforce". The downturn of an organization is assumed to be largely affected by the earned profits. Again, further

analysis will be conducted later in this section. The resulting total workforce, from hiring, firing and quitting, will impact on the total human resource costs ("Total HR costs") imposed on the organization (Sherwood 2002). The costs are calculated by multiplying the "Total workforce" with the "Average HR cost per employee". In other words, the more people hired, the more expenses are needed in terms of salary, and benefits. And during organization financial crisis, firing the employees can reduce the costs.

The "Total workforce" is the summation of both new and experienced staffs, and the workforce is subjected to be transferred in and out during project development. The "Workforce transfer out of $project_{A,B}$" returns the workforce from individual projects back into the "Total workforce" resource pool when there is excess workforce or the project has been completed, and the "Workforce transfer into $project_{A,B}$" acts in an opposite direction, i.e., moving people from the resource pool to each individual project when needed. These two variables are further explained in the Planning model.

In Figures 3.12 and 3.13, the Human Resource Model is extended to describe the "Hiring" and "Firing" policies in details. These policies are very much affected by the "Profits" earned in the organization. The factors that can improve the "Profits" are the "Number of projects" and the "Average profit per project". Of course, there are other factors that can change the earnings, but to keep the model simple, we only focus on the direct factors as stated above. Hence,

Profits = (Num of projects * Ave profits per project) – Total HR cost

As the "Total HR cost" rises due to more workforce hired, the profits may drop if the costs are not justifiable. The "Total HR cost" includes all sorts of costs of employment, like salaries, benefits, departmental spending, employment taxes, and the larger these costs, the smaller the "Profits" (Sherwood 2002). Nevertheless, these costs may have little impact on the profits if the organization has superior

management efficiency, excellent software quality and high workforce productivity to handle a large number of profitable projects. Due to the limited scope of this research, the profits stated in the model are only subjected to these three factors as described in the above equation, and thus confines the capability to model the entire financial behavior of the organization.

If the organization intends to improve the productivity of its employees by offering training, the "Amount of training given$_{A,B}$" may reduce the profits in a short term, but may improve the staff's ability in the long run. For instance, a study of the training practices of 3,000 businesses as related to productivity, conducted by the National Center of Educational Quality and the Workplace at the University of Pennsylvania and the U.S. Census Bureau, revealed that money spent on training produced twice the gain in efficiency and productivity as that spent on tools and machinery (Brooke 1996). In other words, although the cost of training programs is high, the 'cost' of a poorly trained work force is higher (Hanson 2000). The financial status of an organization may also pose an impact on the amount of training provided to its employees. When the organization is at a financial loss, the training may be reduced from the original scheduled allocation, and conversely may be increased when the financial status is healthy. As a result, the productivity of the workers may be affected indirectly by the financial status of the organization. Further explanation can be found in the Productivity model.

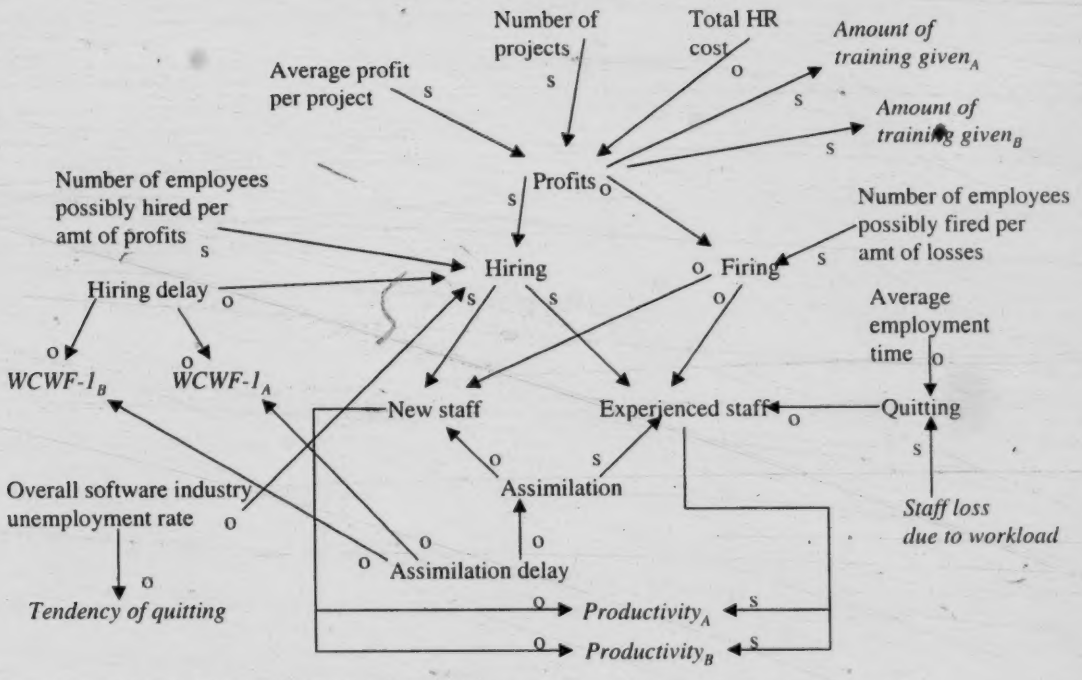


Figure 3.12: Human Resource Model (Extended) in causal-loop diagram

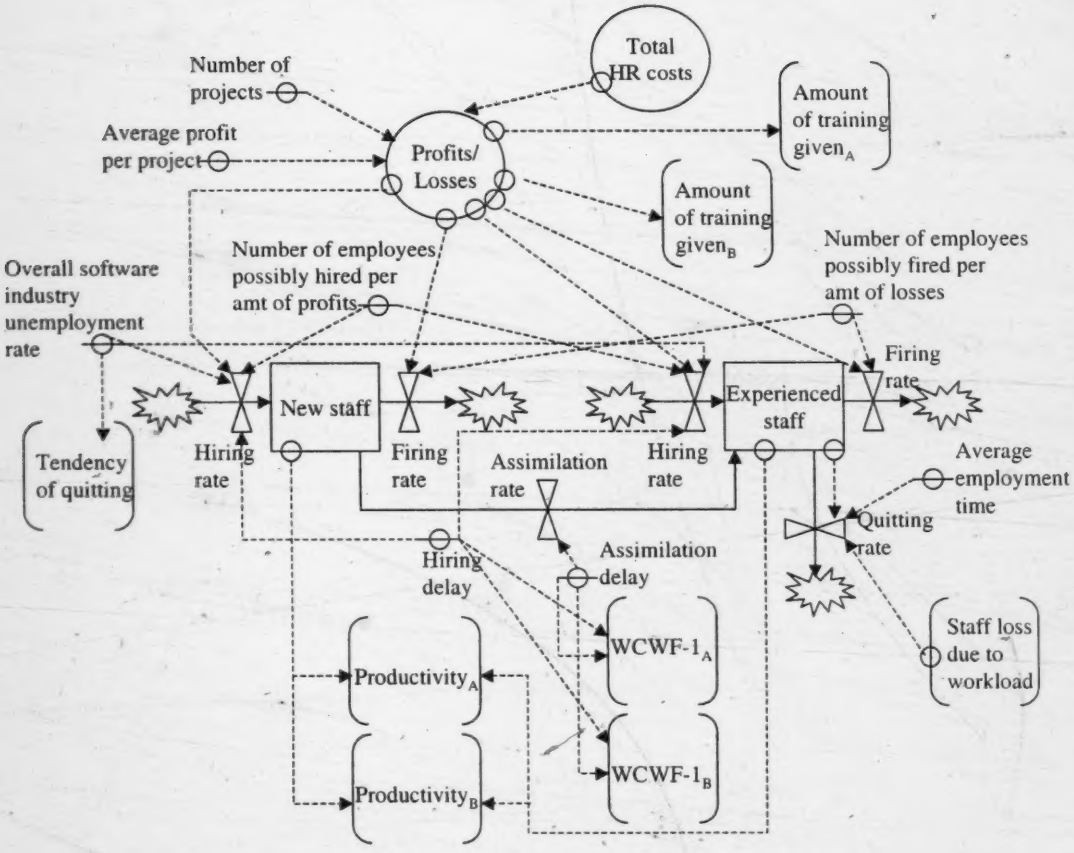


Figure 3.13: Human Resource Model (Extended) in flow diagram

In the hiring policy, we assume that an organization may not hire additional worker even the need arises throughout financial downturn. During the time of crisis when the organization is not making money, the organization may not hire any additional workers, and this effect is represented by the "Number of employees possibly hired per amt of profits". In other words, when there is no profit, no worker will be hired, but as the profit increases, the workforce level will be allowed to grow based on the number allocated. For example, for every $100,000 of profits per month, the organization is allowed to hire a maximum of 10 workers per month. If the profits drop below that number, the human resource department may freeze hiring at the moment.

In addition to the profits, the hiring process may also be affected by the "Hiring Delay" (Abdel-Hamid 1991), and "Overall software industry unemployment rate". If the overall unemployment rate is low, the overall demand of the workforce is high, and this may result in difficulty in hiring, and vice versa. The delay time of hiring is often felt in the professional workforce, especially when the special skilled people are needed, and may not be available within a certain period of time. The following table summarizes the relationships.

Overall unemployment rate	Multiplicator	Hiring Delay (Nominal = 30 days)
Low	1.5	1.5*30 = 45 days
Medium	1.0	1.0*30 = 30 days
High	0.5	0.5*30 = 15 days

Table 3.3: Effect of unemployment rate on hiring

The exact values of the multiplicator and the nominal hiring delay may certainly vary in different organizations, countries, types of professionals hired, and economic standing. The essential element here is to capture the effect of unemployment rate on the hiring delay often experienced in the software industry.

As for the firing policy, when the organization is losing money with negative profits, the workforce level may shrink as the "Number of employees possibly fired per amt of losses" increases. For example, for every $100,000 of losses per month, the organization is allowed to possibly fire 10 workers per month. Firing a worker would yield higher expected returns, as the firm is reducing its future losses for a period of time (Booth 2002). If the "Workforce variance" shows an excess of workers in this period of time, the workers may be fired. The number of people fired is determined by the minimum number between the "Workforce variance" and the "Number of employees possibly fired per amt of losses". In other words, even if the company is making losses, the workers may be retained if they are needed in the project development, but may be fired once there is an excess of workforce.

With reference to Figures 3.12 and 3.13, the other factor that causes a change in workforce level is the turnover issues. The turnover is captured as the "Quitting rate" in the model. Referring back to the Workload and Exhaustion model, the turnover issues are also dependent on the overall unemployment rate in the industry. People tend to stay longer despite increasing workload when the unemployment rate is high, and may choose to leave sooner when there are abundant job opportunities. In this model, we assume no turnover among the new staff since it is unlikely for a new recruit to resign during the assimilation period. Hence, the organization may suffer a great impact from the experienced workforce when the turnover rate is high. The average employment time is obtained when the workload is considered nominal. Thus, when the workload exceeds the nominal, the turnover rate may rise above the average level.

As stated before, the "Total workforce" is assumed to consist of two workforce levels, namely "New staff" and "Experienced staff". In order for the new staff, without the necessary experience, to become experienced staff, they need to go through an assimilation period ("Assimilation delay") whereby they get accustomed to the organization's unique mix of hardware, software packages, programming techniques, and project methodologies. Hence, the process of assimilation may reduce the "New staff" level and increase the "Experienced staff" level. The effect of assimilation delay on "WCWF-1" will be explained later in the Planning model.

For each level of workforce ("New staff" and "Experienced staff"), it has an equivalent experience level, which is not shown in the diagram. A worker's productivity depends on the experience, which is technology-specific (Helpman 1999). The ratio of these two experience levels determines the weighted average productivity. For example, in an organization with 50% new staff and 50% experienced staff, the weighted average productivity is calculated to be 10 tasks/man-day. If more new staffs are hired, then the ratio of new staff is higher than the experienced staff's. This differential will reduce the weighted average productivity to be less than 10 tasks/man-day. The opposite is true when the workforce mainly

consists of experienced staffs and the productivity will increase. Further explanation on the weighted average of productivity will be carried out in the Productivity model.

3.2.2 Individual Project Models

Individual project models consist of an Effort model, a Productivity model, a Control model and a Planning model, that are distinctly different from other projects. The notation used in this section is different from the previous section. As there are more variables that are unique to each project than "shared" variables mentioned in the following sections, a subscript "s" is used only for variables in Shared Models, like "$Workloads_s$". If there is no subscript indicated in the variable, then it represents a component from the Individual Project model.

3.2.2.1 Effort Model

Each project has its own requirements on development effort. This model provides some common factors that influence effort determination, and describes how this effort can be adjusted throughout project development as a progress indicator.

Referring to Figures 3.14 and 3.15, the "Effort needed" is the total manpower required to complete the tasks in a project. A task is a unit for sizing up a software product, and can be any arbitrary unit by which we measure a software project size, such as lines of code, function points, modules, and input/output files (Abdel-Hamid 1991). The size of the "Tasks" is first determined by the estimated "Project size". In addition, "Project complexity", coupled with "Project Size" may also change the effort estimated by a certain degree (Garmus 2002). For example, a project with simple user interface may require less effort and tasks for validation and verification, as compared to a project with complex multimedia and virtual reality features that may require more tasks for integration and testing. Furthermore, the "Requirement volatility" may affect the effort needed, as the customers change the requirement, the size of the project may vary as well. Based on some simulations, it is proven that high software requirements volatility is extremely effort consuming, and any investments in system engineering to stabilize it would pay off well (Pfahl 2000). The result is not just some minor adjustments in specification documents, but may affect the entire design and implementation of the software. For example, due to a requirement modification, the development team may have to add (or throw away) a certain

number of features to the existing design. The more volatile the requirement is, the more tasks are needed to make the corrections. An example should help to clarify the combination effect of these three factors on "Tasks". Assume we have a standard of reference, i.e., a completed project that has known values for project size, the volatility of requirement and project complexity. When there is a new project to be developed concurrently, we compare it with this standard of reference. Assume that the initial project size estimate of the new project is 4000 Delivered Source Instructions (DSI), and it is expected to be twice more complicated than the standard project. Hence, the factor "Project complexity" is 2. And, since the new project is almost as volatile in requirement change as the standard project, then the factor for "Requirement volatility" is 1. Combining these three factors, the number of "Tasks" for the new project is 4000*2*1 = 8000 tasks, with reference to the standard project.

Subsequently, the number of "Tasks" determines the "Tasks remaining" in Control model, "Workloads" and "Effort needed" in the project. Based on the American Heritage Dictionary, "Workloads" is "the total amount of work assigned to, or done by, a worker or unit of workers in a given time period", and so it is different from the "Effort needed". Effort is generally measured in man-day, man-month or man-year, and in our case, the unit of effort is set at man-day. Effort mainly depends on the number of "Tasks" and "Productivity". Productivity is usually measured in "tasks/man-day" (Pressman 2001). So, "Effort needed" is calculated by dividing the number of "Tasks" by "Productivity". The higher the productivity is, the lower the effort is needed. "Job size adjustment in man-days" comes into effect during the project lifecycle when the perceived effort needed to complete the project is different from the remaining planned effort needed or "Man-days remaining" (Abdel-Hamid 1991). As a result, the job size is required to be adjusted, and the differences are added or subtracted from the "Effort needed". When the "Effort needed" has been adjusted, we also need to regulate the "Man-days remaining" accordingly so that the new remaining planned effort is identical to the perceived effort still needed. Further explanation about "Man-days remaining" will be provided in the planning model.

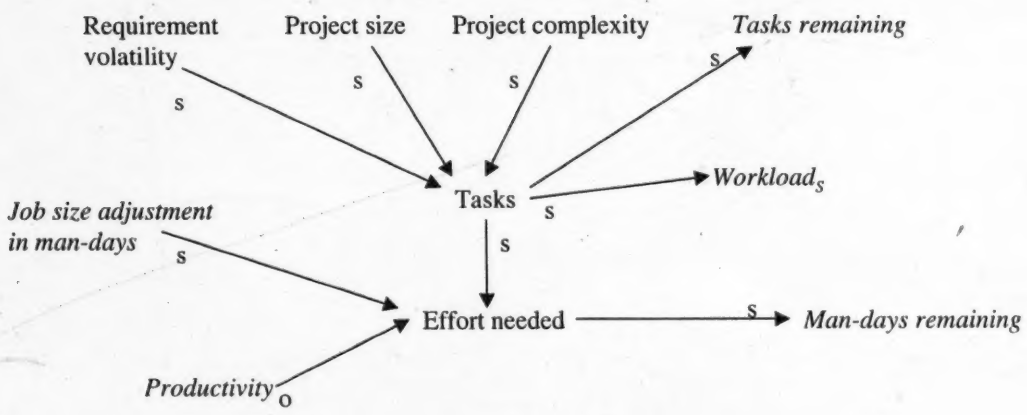


Figure 3.14: Effort Model in causal-loop diagram

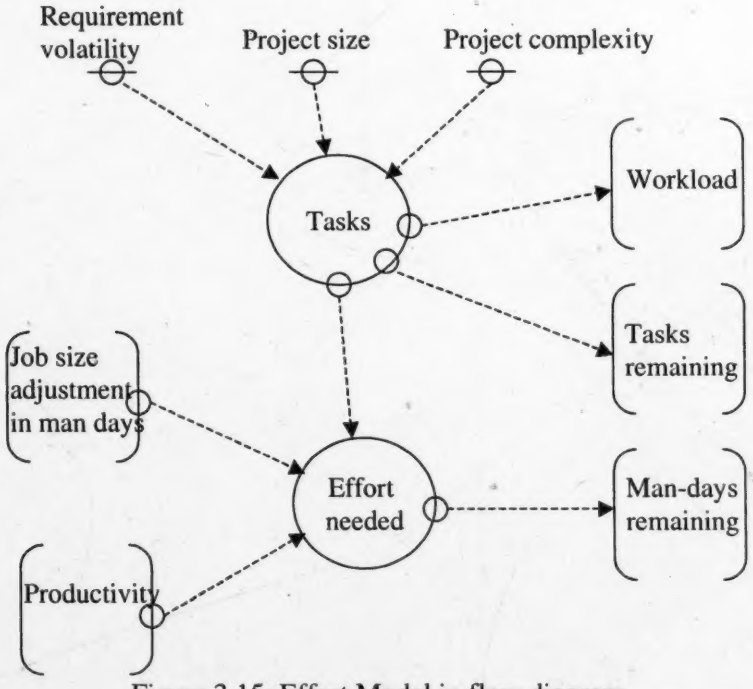


Figure 3.15: Effort Model in flow diagram

3.2.2.2 Productivity Model

Software productivity is a measure of the rate at which software products are being produced. Generally, it is measured in terms of lines of codes, function points, object points, and tasks per man-day. In our case, the unit is set at tasks/man-day.

Productivity can be affected by various factors, and the following model illustrates some of them.

In Figure 3.16, there are few factors that may influence the "Productivity". Two of the factors, "Actual Fraction of a man-day on project" and the workforce level ratio ("New staffs" and "Experience Staffs"), have been explained respectively in Sections 3.2.1.1 and 3.2.1.2 in general. Referring to Figure 3.17, the mixture of new and experienced staffs poses an effect on the productivity. To evaluate this effect, we need to introduce the nominal productivity parameters, one to represent the nominal productivity for new staffs and the other one to represent that of the experienced staffs. The parameters are assumed to be constant throughout the project development. The mixture ratio of the workforce will determine the productivity by using a weighted average of the two parameters. For example, we choose the "Experienced staff's nominal productivity" as a reference and define it as 10 tasks/man-day. Then, the "New staff's nominal productivity" is determined relative to it. The value may range from one-third to two-thirds less than that of the experienced staffs, obtained from various interviews and literatures (Abdel-Hamid 1991). However, this value can be different in organizations, and should be adjusted accordingly. Assuming, its value is set at 5 tasks/man-day, i.e., relatively speaking, the new staffs are about 50% less productive than the experienced staffs. If the mixture of workforce experience at this point of time is 60% experienced staffs and 40% new staffs, then the productivity is calculated as the weighted average of these parameters, i.e., (0.6*10+0.4*5) = 8 tasks/man-day.

Figure 3.16: Productivity Model in causal-loop diagram

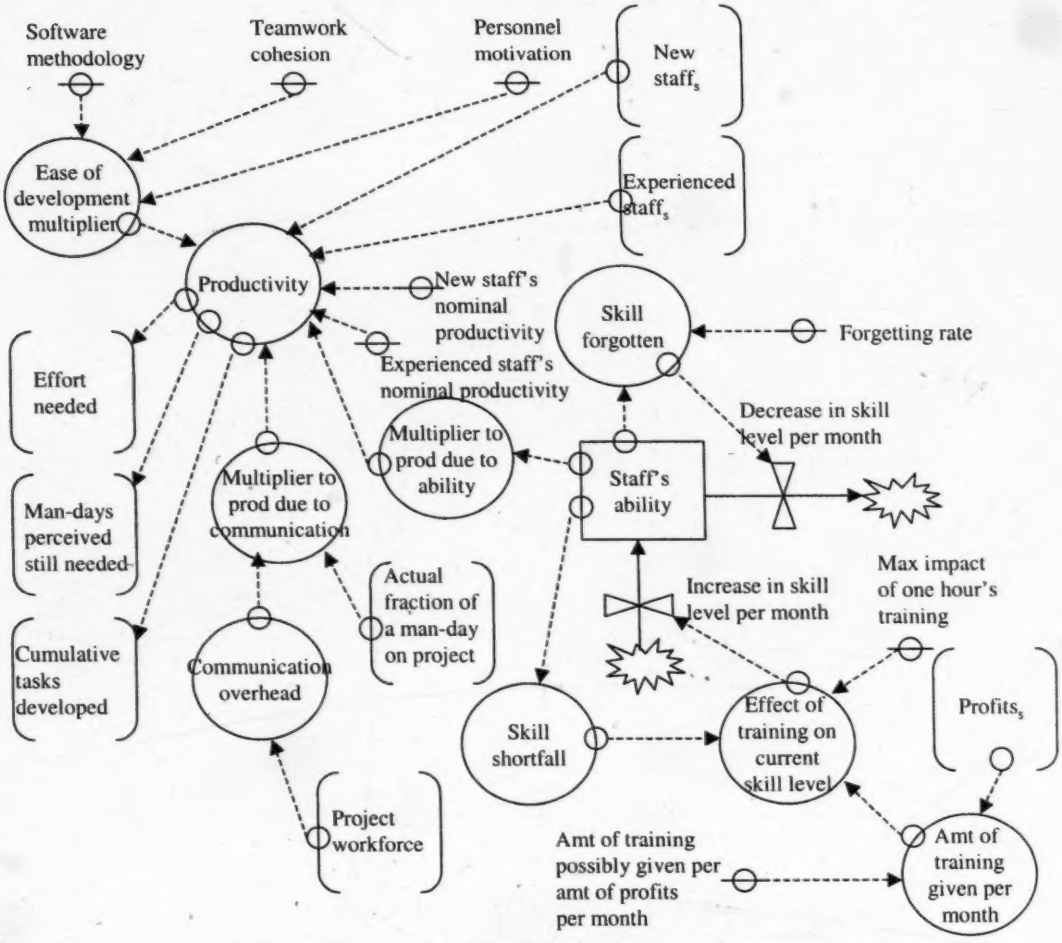


Figure 3.17: Productivity Model in flow diagram

One of the other factors is "Communication overhead". There is a negative effect of involving too many workers in a single project. The overhead, like the meetings held to coordinate work, the interference due to miscommunication, and the measures to solve these communication conflicts, will impair the productivity of individual, as more time is needed to spend on correspondence. When the team size is small, there is not so much communication overhead, and it provides productivity advantage (Jeffery 2002). When the size of development team is growing, the communication required to sustain the daily functions of the team may grow as well. Ignoring this overhead cost may reduce the net benefits gained from a healthy communication channel by (Long 1995):

- becoming slower to respond to the market
- less efficient at assimilating change on many fronts
- less effective in capturing and deploying knowledge
- and less efficient overall

It is widely held that communication overhead increases in proportion to n^2, where n is the "Project workforce" in a team, and the relationship is shown in Figure 3.18 (Abdel-Hamid 1991). For example, referring to Figures 3.17 and 3.18, if the "Actual fraction of a man-day on project" is 0.6, i.e., an employee allocates about 60% of his working time on this project each day, and if the total project workforce, n, is 30, then the multiplier to productivity due to communication overhead is 0.6*0.5=0.3. In other words, the actual productivity is only 30% of the potential productivity when taking into account of communication overhead.

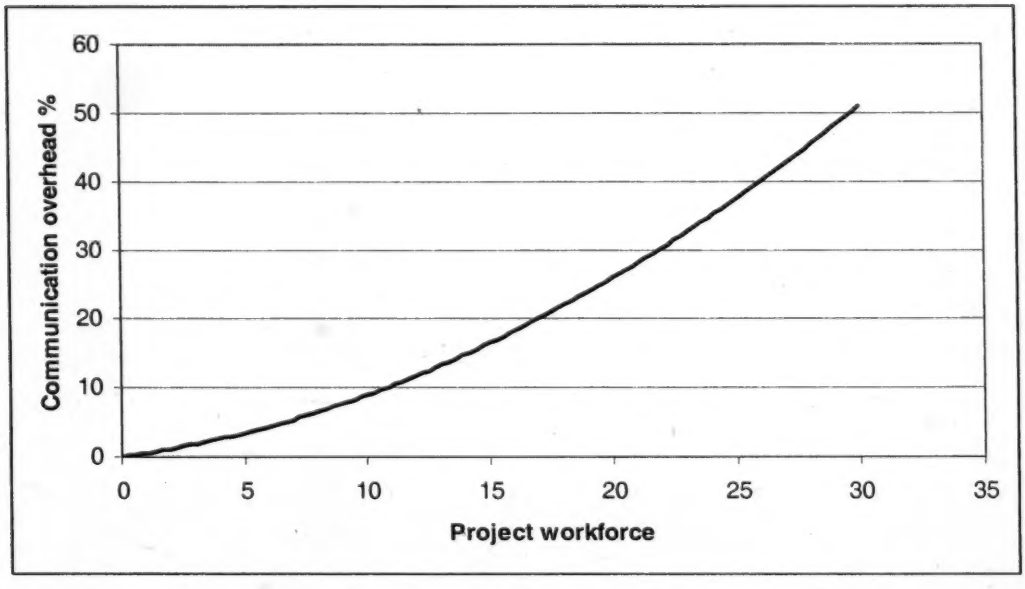


Figure 3.18: Communication Overhead

The fourth factor deals with "Staff's ability", which is obtained mainly through training. Experience and training are two different types of human capital. Both of them increase the productivity of the labor force, but they operate through different

mechanisms (Helpman 1999). Training increases the set of technologies that a worker can operate, whereas experience increases the productivity of a worker with a given technology. Training for the employees in job related fields would enhance their knowledge and in return, enhance their ability to improve the productivity. When the organization faces declining profits, the management needs to meet the training requirement cost efficiently (Henderson 2001). As shown in Figure 3.17, the current "Profits$" of an organization may often affect the amount of training given. During bad time, we assume that the management tends to cut cost on the training provided. For example, if we set "Amt of training possibly given per amt of profits per month" to be 5 hours/month per monthly profits of $10k, and if the current profits are $5k, then the amount of training this month is 2.5 hours. This relationship is assumed to be linear with respect to the amount of profits, but there may exist a maximum threshold of allowable training given when the profits keep rising. The "Amount of training given" then has an effect on the employees' skill level. Initially, an increased of training will raise the skill levels in a linear manner. However, as the average skill level rises, the potential for further benefits from training is progressively reduced (Warren 2002). This is shown in the "Effect of training on current skill level". This effect is best illustrated with an example in Figure 3.19. Assuming at the initial stage of the project, the staff's ability is considered 70% of the highest achievable ability upon completion of training, i.e., the highest ability is set at 1, and the current level is 0.7. The "Skill shortfall" is then 1-0.7 = 0.3. The "Effect of training on current skill level" is a product of the skill shortfall, the amount of training given per month (set at 2hr/month), and the maximum impact of one hour's training (set at 10%). At the beginning of the project, this effect is 0.3*2*0.1=0.6. In other words, at this moment, the staff's ability can be increased by 6% by the end of the first month, and then there is a steep increase in the first few months. However, as several months have passed by, the potential for further increase in staff's ability reduces and the shortfall has progressively dropped. As a result, the effect of training on the skill is decreased in a similar fashion. This is a result of a balancing feedback that seeks to achieve a goal, which is 1 in this case. When the value is below the goal, the loop pushes its value up towards the target.

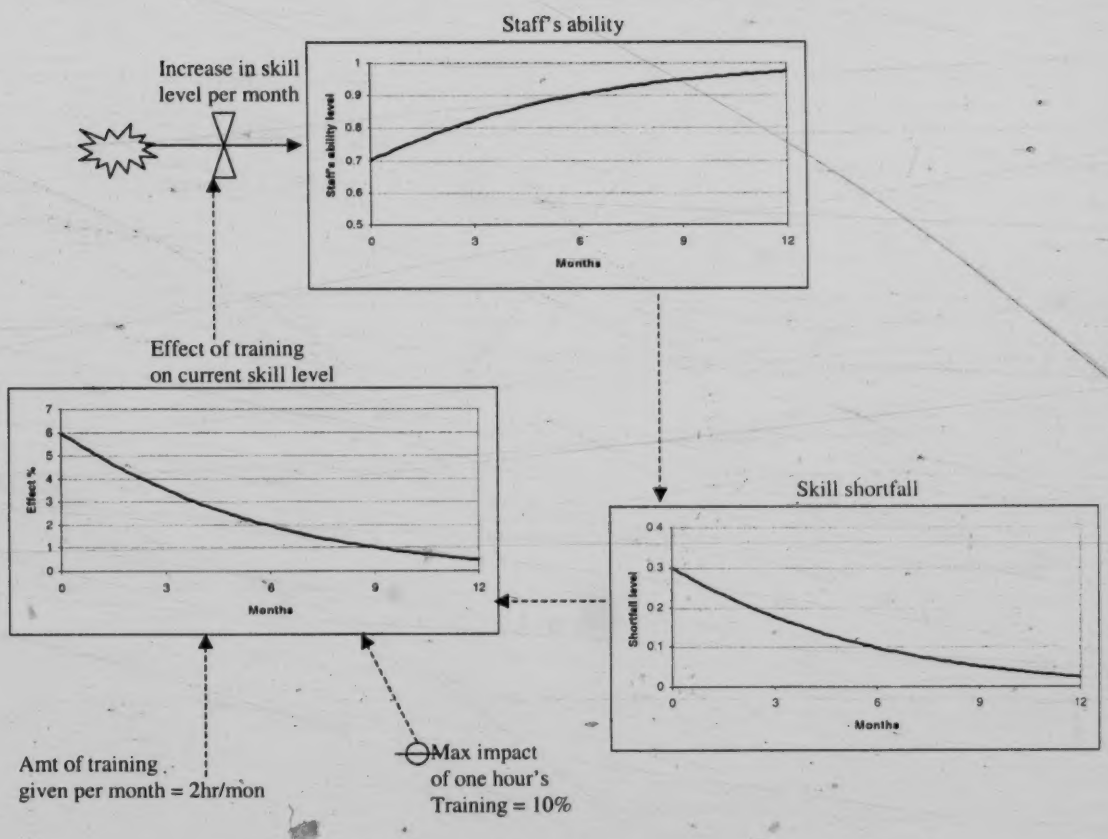


Figure 3.19: Increase of staff's ability due to training

As time passes, if the related skill and ability are not utilized, the worker may forget part of those skills or "Skill forgotten" (Warren 2002), and thus may reduce the ability to improve the productivity. Referring to Figure 3.17, if the "Forgetting rate" is 5% per month, the skill forgotten shows a decrease of staff's ability of 5%. The decline of skill level may not be permanent, and we assume that the ability may decline until the starting point of the level when there is no training provided, i.e., 0.7 based on the example in Figure 3.19. Although the relationship between the staff's ability and productivity is obvious, it is hard to be quantified. An increase of staff's ability by 10% does not necessarily cause an increase of productivity by an equal amount. If the staff's ability is enhanced but is only useful in a small limited project domain, the effect on the productivity on the entire project may not be as significant. Hence, the multiplier to productivity due to staff's ability is subjected to the evaluations of different situations and cannot be determined at this point of time.

The productivity of a project is also affected by the "Ease of development". This includes "Software methodology" used to develop the software, "Teamwork cohesion" and "Personnel motivation". The multiplier to productivity due to ease of development is a product of these three factors.

There are a variety of software methodologies used in the development of software. The traditional method is the waterfall approach (Pressman 2001), which requires a development phase to be completed before the next phase is started. Recently, due to the emphasis on agility and the pressure of time-to-market, some software organizations are moving into light and agile methods that may require shorter life cycles and potentially result in higher productivity. For example, an observation done on 14 software firms shows that the productivity of the software engineers using agile methods improves 15% to 25% on average in lines of codes, compared to the published industry benchmarks, without sacrificing any quality in the products (Reifer 2002). In another study on the web-based application software, the Extreme Programming increases the productivity by 66% in lines of code, by surveying nine full-time software developers (Maurer 2002). The above observation and study show a change of productivity with different methodologies, and the percent increase in productivity varies from cases to cases, and should not be taken literally. The data are constantly changing as new methods have been discovered or old methods have been improved. As a result, no definite values can be determined without considering the standard methodology used before in the organization and how the current new or improved methods compared with that standard.

As most projects involve a group of people, the effective patterns of teamwork show workgroup cohesion, and collective capability that are significant in the productivity improvement (Lakhanpal 1993). The practices may involve collocation, like pair programming (Williams 2000), war room (Teasley 2002) and joint application design workshop or JAD (Davidson 1999), and etc. In one study, a comparative statistics on the productivity measures were conducted on six teams, and the pilot team using the

war room teamwork structure shows 100% improvement in function point per staff month compared to the company baseline, and 160% improvement compared to the industry standard (Teasley 2002). By encouraging team cohesion, it provides generally the horizontal communication between team members so that the team unity and coordination is facilitated (Teasley 2002), and thus, presumably, the productivity may increase in a certain extent. However, no sufficient empirical data has been provided to date. It is suggested a similar approach employed by COCOMO II to correlate effort and team cohesion be used (Boehm 1995). Different characteristics of team cohesion can be identified, such as experience of members work in teams, willingness to accommodate others' objectives and the extent of shared vision. Then, the characteristics can be associated with the productivity based on available industrial data.

Motivation is a measure of enthusiasm to work and having positive views about the job initiatives. Software engineers are found to have unique characteristics (Sharp 1999), and so it may be necessary for the managers to understand them in order to improve their productivity. There are a number of motivators for software engineers, such as visibility of success, resource availability, management commitment, ownership and reward schemes (Baddoo 2002). In general, highly motivated people may ease the development of the project by having a higher productivity in work, as they tend to work longer and harder than others (White 1999). However, more extensive research needs to be done due to a shortage of empirical data. An approach similar to team cohesion's can be carried out by identifying the different motivators of software developers, and then evaluate the impact of each motivator on the productivity.

Finally, using the "Productivity", the "Man-days perceived still needed" and "Cumulative tasks developed" are calculated in the control model as elaborated in the next section.

3.2.2.3 Control Model

In the control phase of a project, it may be essential to assess the progress and determine the remaining effort perceived still needed to complete the tasks. The progress in a software project is measured by the effort spent, tasks completed or both (Abdel-Hamid 1991). This model utilizes the Effort model to determine the remaining tasks and effort required to complete the development. The effort still needed is then compared with the remaining effort, and the variation of effort is then used to decide if there are any man-day shortages or excesses. Once the assessment has been made, the project parameters and the workforce may be altered to compensate for the change. For example, if the project is behind schedule, the work rate needs to be boosted in order to maintain on-time delivery. The boost work rate is then converted to an increased productivity. Nevertheless, in case the work rate cannot be increased sufficiently to cope with the schedule, then the total effort needs to be readjusted which would delay the development schedule. The following diagrams, Figures 3.20 and 3.21 are obtained partly from Abdel-Hamid's software development productivity sub-sector that describes these scenarios.

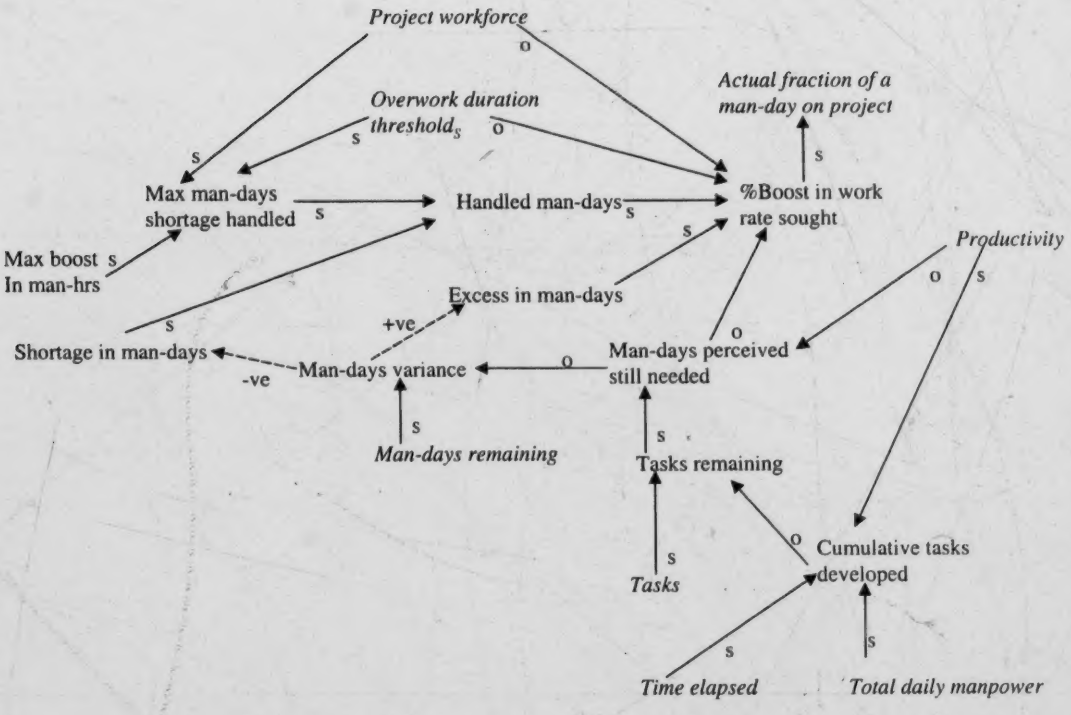


Figure 3.20: Control Model (General) in causal-loop diagram

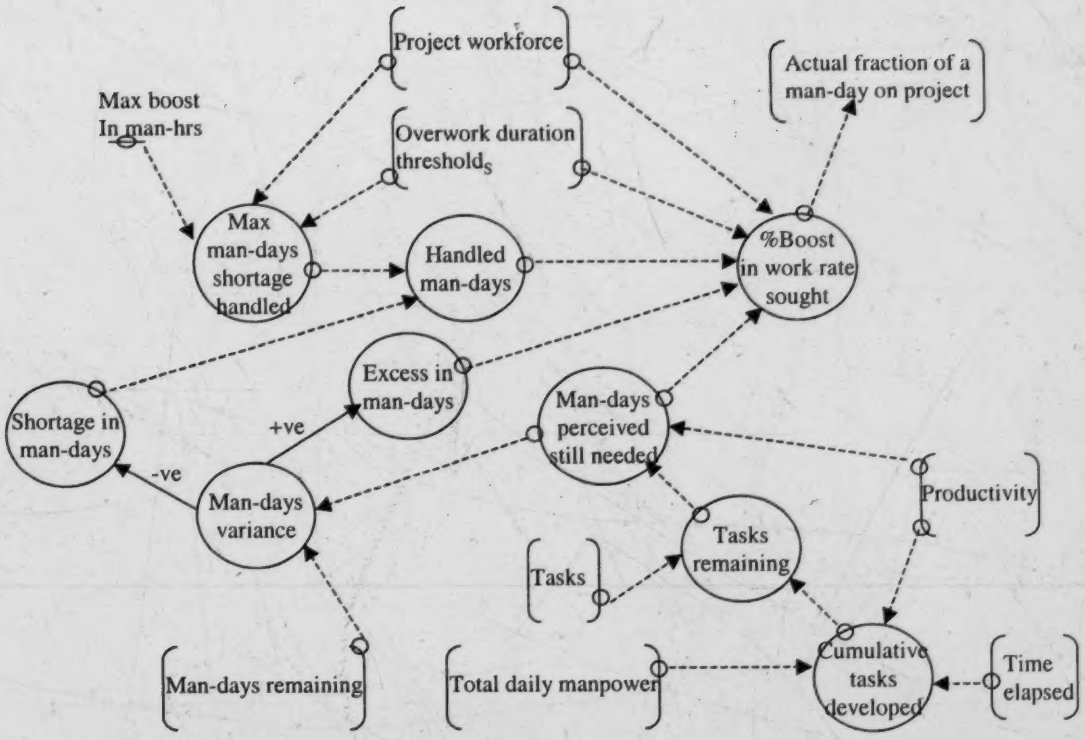


Figure 3.21: Control Model (General) in flow diagram

As the development of the project progresses, more and more tasks are developed, and are accumulated in the "Cumulative tasks developed" which is calculated by multiplying the "Productivity" with the "Total daily manpower" and "Time elapsed". For example, a project that has been developed for 2 days with a productivity of 10 tasks/man-day and daily manpower of 10 man-days/day, the cumulative tasks developed is 200 tasks. "Cumulative tasks developed" measures the progress of the project, i.e., at the beginning of the development, its value starts with 0, and when its value equals "Tasks", which is the total number of tasks in the project, the project is considered complete. Then, by subtracting the "Cumulative tasks developed" from the "Tasks", we get the "Tasks remaining" which describes the remaining tasks waiting to be developed. To convert the unit of "Tasks remaining" in tasks to man-days, it is then divided by the "Productivity", and the result is called "Man-days perceived still needed".

"Man-days perceived still needed" is the total effort perceived still required to complete the project in man-days. "Man-days remaining" is the remaining actual effort needed to complete the project, and is determined in the Planning model. By comparing the "Man-days perceived still needed" with the "Man-days remaining", we can find the discrepancy that is the "Man-days variance". This is the difference between the actual and planned effort, i.e.,

Man-days variance = Man-days remaining – Man-days perceived still needed

If a positive value is obtained, it means the project is way ahead of schedule, or in simple term "Excess in man-days". However, if the project is behind schedule, the negative value of "Man-days variance" is called "Shortage in man-days". Two dotted lines in Figure 3.20 (and two solid lines in Figure 3.21) are used to represent two possible cases for "Man-days variance", i.e., one for "Excess in man-days" and one for "Shortage in man-days".

Before going in details about excess/shortage in man-days, we need to understand "Max man-days shortage handled". It is a threshold that the employees are willing to handle overtime in man-days, equivalent to the "Overwork duration threshold$_S$" as explained in the Workload and Exhaustion model. It is determined by the product of three variables: the "Overwork duration threshold$_S$", "Project workforce", and "Max boost in man-hrs" (Abdel-Hamid 1991). For example, a project team of 10 workers is willing to work overtime for only 10 days, i.e., their overwork duration threshold is only 10 days. During that period, they could boost their work rate by as much as 100%, that is double their normal working hours everyday for the next 10 days. Then, the maximum overtime they can handled or "Max man-days shortage handled" is 10x10x1=100 man-days of extra effort to be used to boost the work rate.

When there is a "Shortage in man-days", i.e., when the project is perceived behind schedule, the "Handled man-days" is determined by the minimum value of two variables: the "Max man-days shortage handled" and "Shortage in man-days". When the latter is smaller than the former, it means that the workforce, through working

overtime, can absorb whatever shortage of effort needed to complete the project without affecting the project completion schedule. However, when the latter is greater than the former, the shortage of effort exceeds the workforce's capability to handle the maximum overwork effort. So, the workers are only willing to work up to the maximum man-days threshold, and the remaining effort that can not be handled within the duration will be used to readjust the schedule which will be explained in the Planning model in the next section.

The "% Boost in work rate sought" defines an enhanced work rate goal in terms of man-days fraction allocated to the project, which equals the value of "Handled man-days" divided by the product of "Project workforce" and "Overwork duration thresholds" (Abdel-Hamid 1991). For example, if 100 additional man-days are to be handled by a 10-person team in 50 days, the percent boost is 100/(10x 50) = 0.2, or in other words, they need to boost their work rate by 20% to accomplish the goal. The boost in work rate is then realized in the "Actual fraction of a man-day on project". For example, if the work rate is expected to be increased by 20% in an 8hr workday having 70% of the time spent on the project, then the actual fraction of a man-day is increased from 0.7 to 0.84.

For the case of "Excess in man-days", when the project is ahead of schedule, the workers may first reduce their work rate before downward adjustments are made in the schedule (Boehm 1981). The reported effort still needed is slightly more than the perceived effort still needed, as the workers tend to under work (Abdel-Hamid 1991). The "% Boost in work rate sought", in this case, shows a reduction in work rate, and is determined by the differential of the reported and perceived effort. For example, if the reported effort or "Man-days remaining" still needed is 4 man-days and the "Man-days perceived still needed" is 2 man-days, then the percent reduction of work rate is 50% or 0.5.

After the above assessment of shortages or excesses of man-days has been done, and if there are any shortages or excesses that are not being absorbed by an increase or

decrease of work rate, the differential will be reported and readjusted in job size (Abdel-Hamid 1991). Referring to the following figures (Figure 3.22 and Figure 3.23), the "Reported shortage/excess in man-days" is determined by subtracting the "Handled man-days" from the "Man-days variance", that is, if the workers can only handle up to 10 man-days of extra effort while the perceived shortage in man-days is 15 man-days, then the reported shortage is 5 man-days. These shortage effort needs to be considered for readjustment of job size.

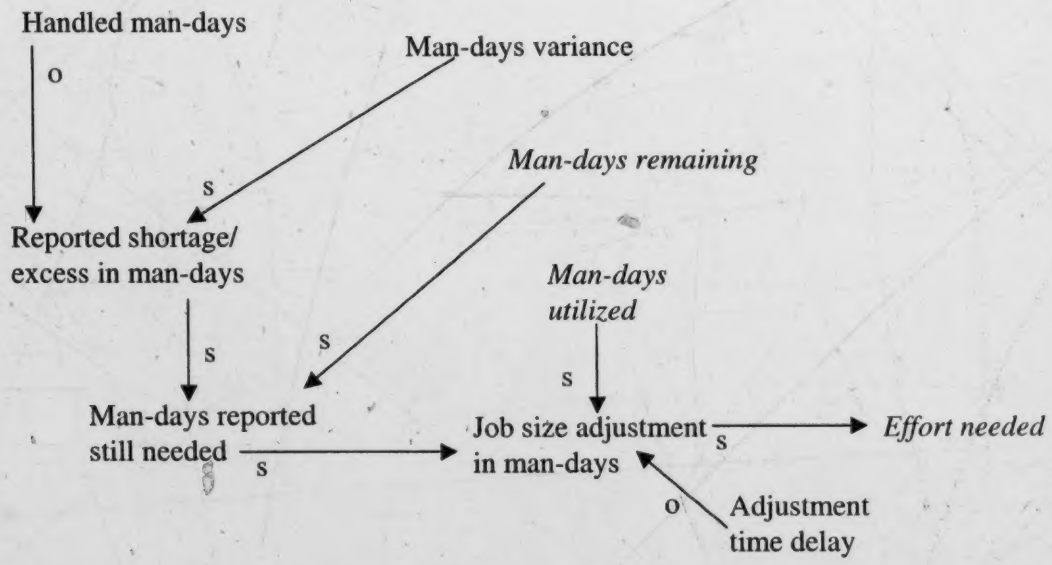


Figure 3.22: Control Model (Extended) in causal-loop diagram

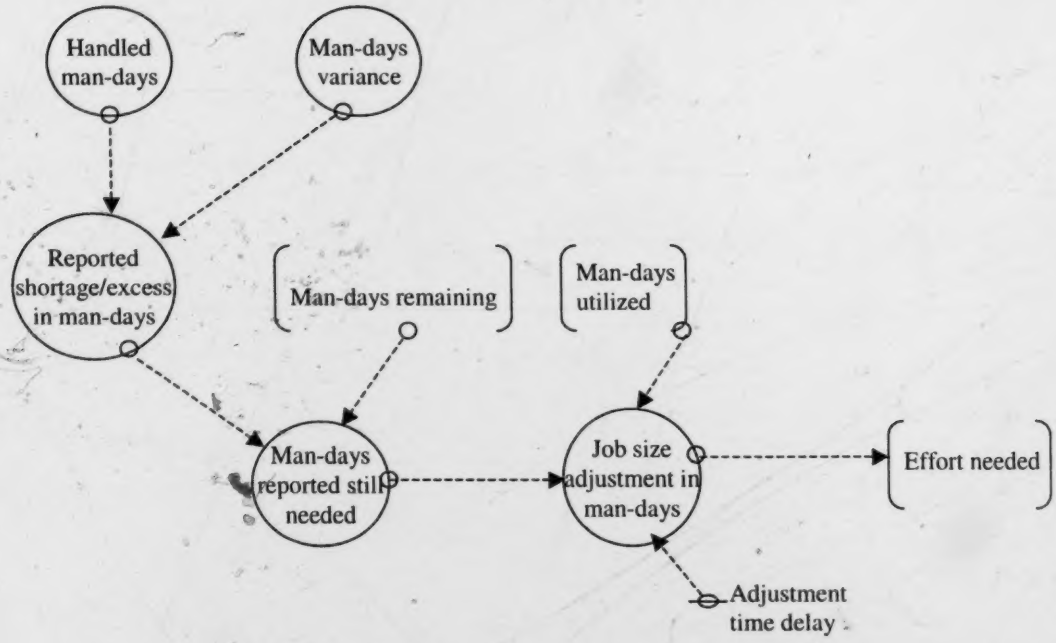


Figure 3.23: Control Model (Extended) in flow diagram

When this value is added to the "Man-days remaining", we get the total "Man-days reported still needed". Then, the job size or effort needs to be readjusted to accommodate this differential. The rate of adjustment is given by the following equation (Abdel-Hamid 1991):

Job size adjustment in man-days = [(Man-days reported still needed + Man-days utilized) – (Effort needed)] / Adjustment time delay

The "Adjustment time delay" is the waiting period for the management to adjust the job size, as the adjustment process is not instantaneous. Such adjustment is then translated into adjustment in schedule or in workforce level or both, as will be discussed in the next Planning model.

3.2.2.4 Planning Model

Planning model provides an initial planning at the start of the project and readjusts the schedule and workforce as necessary when the project is under development. At the beginning of the project, this model provides an initial estimate of the required

workforce level for development by specifying the daily manpower and the project start and end dates. Subsequently, as the project progresses, the workforce is readjusted by monitoring the remaining effort and time elapsed. The regulation of workforce, however, is also dependent on the stability of workforce and schedule, which will be further elaborated. If more resources are needed, new workforce may be transferred into the project, and on the contrary, if excess workforce is found in the project, they may be transferred out of the project. In some cases, if workforce adjustment is insufficient to handle the increased effort, then the final project schedule needs to be rearranged. The remaining section provides the details of the Planning model.

Referring to Figure 3.24 and Figure 3.25, the diagrams are similar to the Abdel-Hamid's planning subsystem (Abdel-Hamid 1991). The parameters of "Man-days remaining" and "Man-days utilized" are included in this model, rather than in other models, as the planning of project involves measurement of progress in terms of effort remaining and utilized. The "Project workforce" is determined in this model as this parameter is considered a part of Individual Project Model, and thus cannot be included in the Human Resource model, which is shared with various concurrent projects.

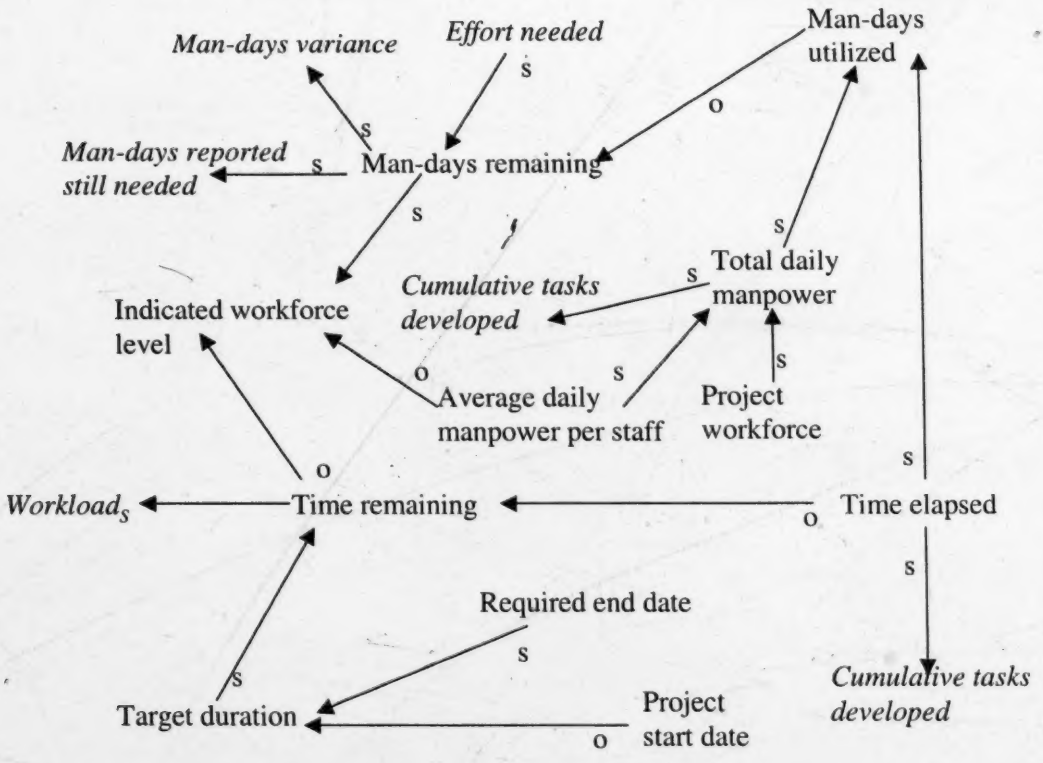


Figure 3.24: Planning Model (General) in causal-loop diagram

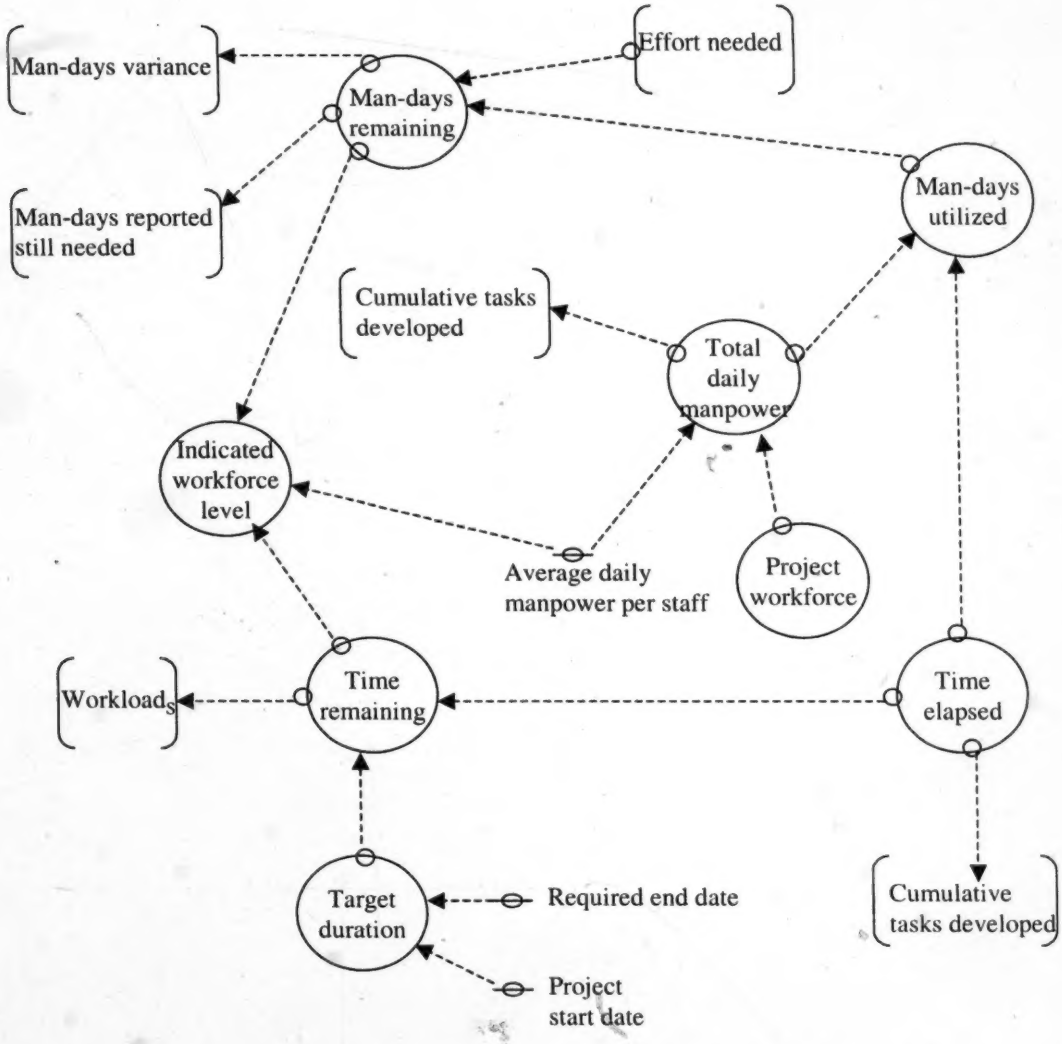


Figure 3.25: Planning Model (General) in flow diagram

"Target duration" represents the number of working days from the start, i.e., subtracting the "Project start date" from the "Required end date". Then, the "Time remaining" is determined by subtracting "Time elapsed" from the "Target duration". "Time elapsed" is the number of days that have been used up since the start of the project. The "Indicated workforce level" is the required number of full-time workers to complete the tasks, and so, in general, it is directly proportional to "Man-days remaining", but inversely proportional to "Time remaining". For example, if the target duration is 100 days and at time 50 days, the value of "Man-days remaining" is

500 man-days, then we determine that the time remaining is 100-50 = 50 days, and the required workforce level is 500/50 = 10 men.

Sometimes, the workers may not work full time on a single assignment, and thus, the "Average daily manpower per staff" may be less than 1.0 (having 1.0 means working full time on the assignment). Hence, the "Indicated workforce level" may be adjusted by dividing it with the "Average daily manpower per staff". For instance, if the workers are only spending 50% of their time in the project, then the average daily manpower per staff is 0.5, and the indicated workforce level becomes 10/0.5 = 20 men.

Then, by multiplying the "Average daily manpower per staff" with the "Project workforce", we can obtain the "Total daily manpower" in man-days/day. Based on the "Total daily manpower" and "Time elapsed", the "Man-days utilized" is a cumulative effort that has been consumed so far since the start of the assignment. Then, the "Man-days remaining" is decided by subtracting the "Man-days utilized" from the total "Effort needed". For example, when 50 days have elapsed, and on the 51st day, the total daily manpower is 10 man-days/day, and the cumulative man-days utilized in the past 50 days is 498 man-days, then by the end of the 51st day, the new cumulative man-days utilized is 498+10 = 508 man-days. If the total effort needed to complete the project is set at 600 man-days, then the man-days remaining is equal to 600-508 = 92 man-days.

The following diagrams, Figure 3.26 and Figure 3.27 show an extended model for the planning phase, which includes readjustment of the workforce and schedule as the project is under development.

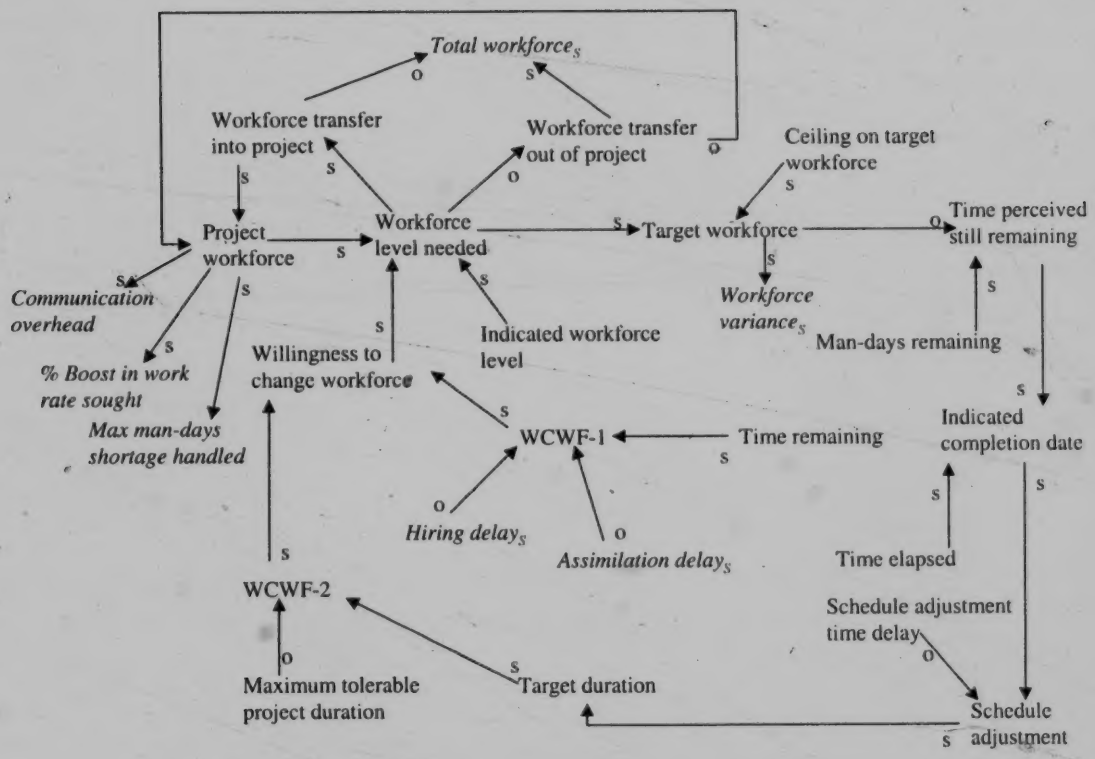


Figure 3.26: Planning Model (Extended) in causal-loop diagram

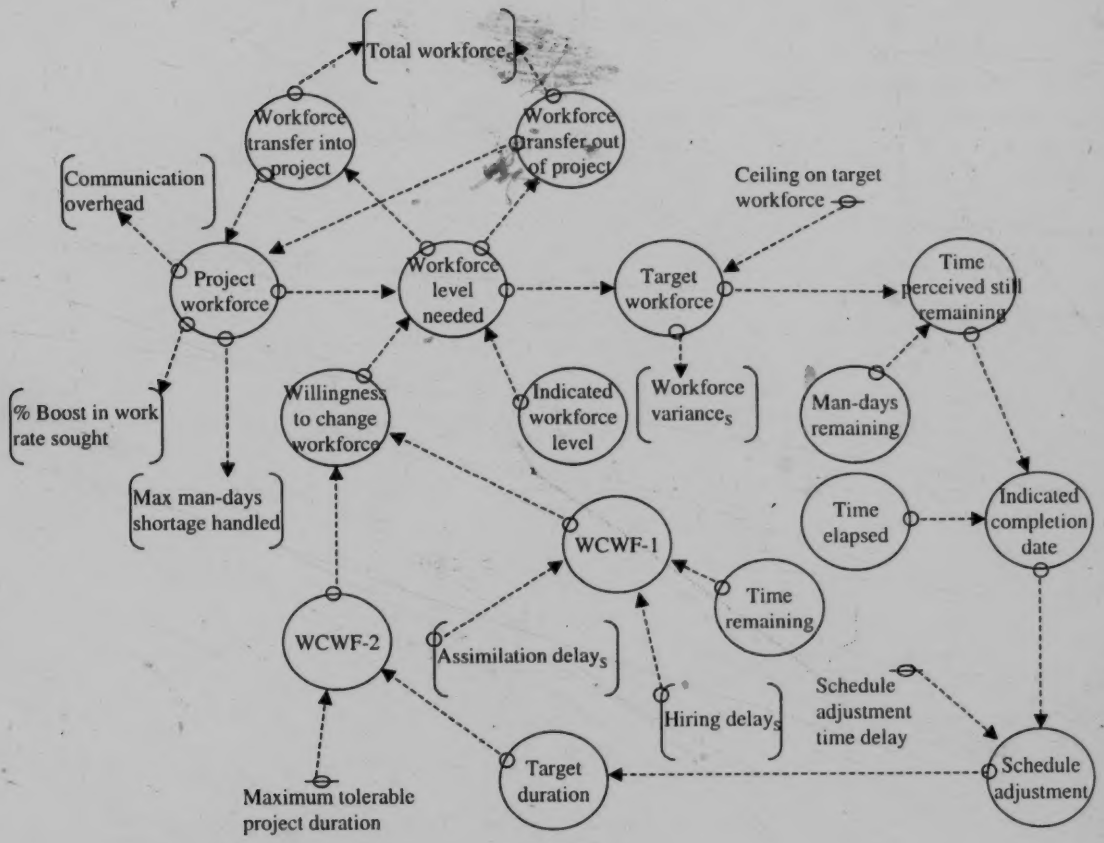


Figure 3.27: Planning Model (Extended) in flow diagram

The "Project workforce" is the number of workforce required in a single project, and it determines the "Workforce level needed" in the planning phase. When the workforce actually needed in a project ("Indicated workforce level") is less than the current "Project workforce", it means that there are more people working in the project than needed. So, some people may be transferred out ("Workforce transfer out of project"), and returned to the pool of "Total workforces", waiting to be assigned to other projects. However, when the workforce needed is more than the "Project workforce", it means that the project needs more people than before, and if there is available workforce in the "Total workforces" pool, then more workers can be transferred into the project ("Workforce transfer into project"). "Communication overhead" is affected by the number of workforce in the project, and in return, reduces the productivity as explained before in the Productivity model. The "Project

workforce" also governs "%Boost in work rate sought" and "Max man-days shortage handled" in the control phase. All these relationships have been explained in their individual models.

Sometimes, the "Total workforces" may not be available since other concurrent projects may have used up the resources, then the "Workforce transfer into project" will equal zero, or in other words, no additional workers can be transferred and more workers may need to be hired. However, the hiring decision is mainly dependant on the hiring policy explained in Section 3.2.1.2, as well as the stability of workforce versus the stability of schedule. The following equation (Abdel-Hamid 1991) is applied when there is a need to hire more people by considering the willingness to change the workforce:

Workforce level needed = Indicated workforce level x Willingness to change workforce + Project workforce x (1-Willingess to change workforce)

"Willingness to change workforce" or WCWF is a variable that assumes values between 0 and 1, inclusive. It weighs the stability of workforce against the stability of schedule. For example, when the project is in the initial stage, the management may be more willing (WCWF=1) to hire more people in order to ensure the project is completed on schedule, i.e., to ensure the stability of schedule. However, as the deadline is approaching, the willingness to change the workforce drops to 0 (WCWF=0), the management is less willing to hire people as there is a hiring delay and assimilation delay wasted to recruit new workers and acquaint them with the projects, and the existing workers may need to divert from their primary and critical tasks to train them. Furthermore, by the time the new workers are familiar with the tasks, the project may have drawn near or even passed the scheduled deadline. This is the golden rule of Fred Brooks (Brooks 1995) that states that adding more developers to a late project will only make it later. Thus, normally in this situation, the organization may prefer to maintain the stability of the workforce without hiring new workers and may have to readjust the schedule to accommodate the delay.

WCWF can be divided into two components, i.e., WCWF-1 and WCWF-2, based on the following equation (Abdel-Hamid 1991).

$$WCWF = Maximum\ (WCWF\text{-}1,\ WCWF\text{-}2)$$

First, "WCWF-1" refers to the stability of workforce. It is a function of "Time remaining", "Assimilation delays", and "Hiring delays" as illustrated in Figure 3.28.

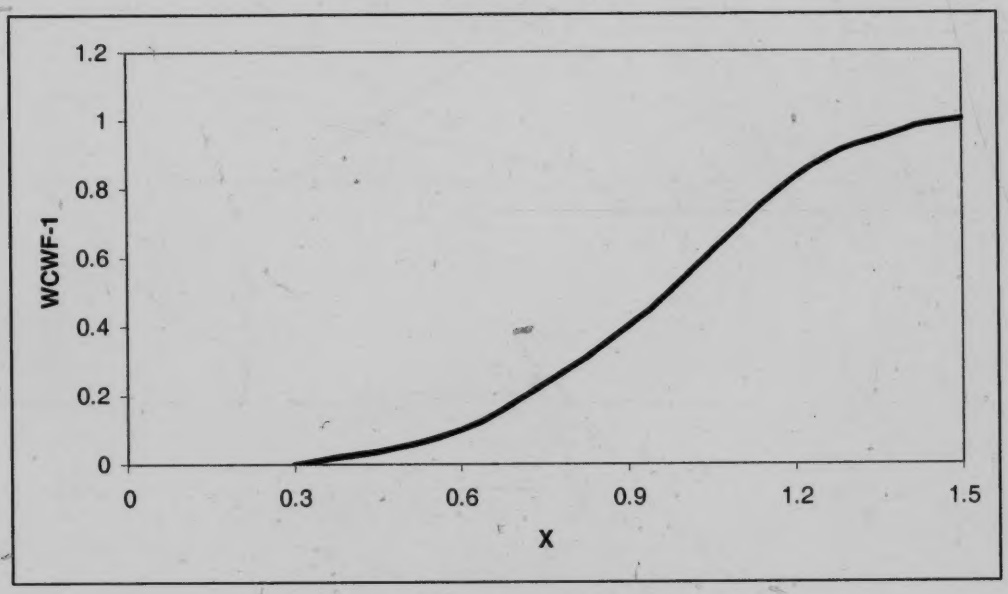


Figure 3.28: General form of WCWF-1

$$\text{Where, } X = \frac{Time\ remaining}{(Hiring\ delay + Assimilation\ delay)}$$

As the project has just started, the "WCWF-1" equals to 1 and as time progresses, the value drops to 0 as the deadline draws close, which is mainly based on Brooks' law. In the early stage of a large project, the time remaining may be much larger than the total delay time required to hire new people and to train them during assimilation period, i.e., X>1.5. Hence, WCWF-1 is equal to 1, and the management would be totally willing to increase the workforce level to suit the project's scheduled

completion date. However, as the time remaining drops further, the trend shows that the resistance to increase new workforce is on a rise. For instance, if the hiring delay is 30 days and assimilation delay is 120 days, then as the time remaining drops below 225 days, the reluctance to hire new people is escalating despite an increasing demand of higher workforce level. When the time remaining drops further till 45 days and below, the hiring rate is set to zero, and the management may have to delay the project's scheduled completion date to accommodate the lack of workforce supply.

The other component is "WCWF-2" which emphasizes on the stability of schedule. Cost overruns due to delays of project are common, and can cost from 100% to 200% of the budgeted costs, and the projects are delayed to the point where the market conditions for which they were designed have changed (Sterman 1992). When the management realized this outcome, they may do whatever necessary to avoid overshooting the maximum project completion date or "Maximum tolerable project duration". Hence, WCWF-2 is a function of "Target duration" and "Maximum tolerable project duration" as shown in Figure 3.29. The latter is a safety buffer, employed by the management to ensure on-time delivery (Abdel-Hamid 1983). The higher the losses when the project is delivered late, the higher the safety buffer. For example, if the maximum tolerable project duration is 100 days, and a 25% safety duration is considered, then the initial target duration is 75 days.

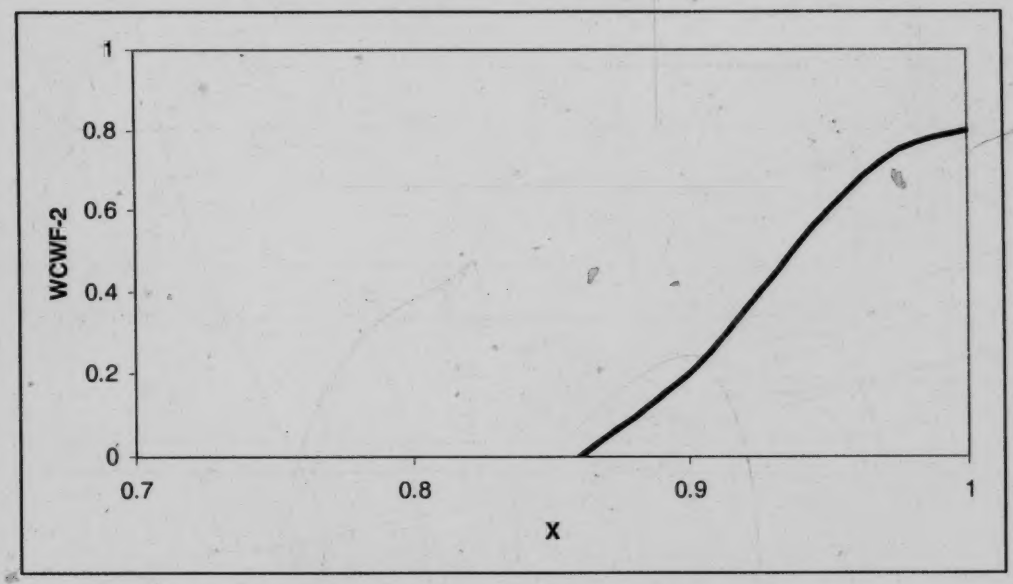


Figure 3.29: General form of WCWF-2

$$\text{Where, } X = \frac{T \text{ arg } et \text{ } duration}{Maximum \text{ } tolerable \text{ } project \text{ } duration}$$

If the project has fallen behind schedule, but the new target duration is still far less than the maximum tolerable project duration, then the willingness to change the workforce is still based on the balancing of both workforce and schedule stability, i.e., WCWF-1 and WCWF-2 may both have effect on the management decisions. However, when the target completion duration is further pushed back until it approaches the maximum tolerable duration, the pressure to finish the project before the tolerable date is mounting to an extent of overruling the workforce stability considerations. In other words, this pressure overrides the workforce stability WCWF-1, and the management may be willing to hire more people to ensure schedule stability.

Referring back to Figure 3.26 and Figure 3.27, the "Target workforce" is set at the value of "Workforce level needed" as long as it is less than or equal to the "Ceiling on target workforce". If not, the "Target workforce" is set at the latter. This ceiling is

the maximum allowable workforce in the project set by the management. "Time perceived still remaining" represents the remaining working days perceived to be needed to complete the project. It is computed by dividing the "Man-days remaining" by the "Target workforce". The "Time perceived still remaining" is then added to the "Time elapsed" to obtain the "Indicated completion date". The latter value signifies the new completion date of the project, and needs to be adjusted in the target duration. The "Schedule adjustment" has a delay ("Schedule adjustment time delay") for the management to realize and react to the adjustment. The new "Target duration" is then adjusted by applying the equation below (Abdel-Hamid 1991):

Schedule adjustment = (Indicated completion date – Target duration level)/Schedule adjustment time delay

For example, if the management decides to hire additional 2 workers for the project development team of 8 workers due to possible schedule slippage, the new target workforce is then 8+2 = 10 workers. Presumably the remaining effort to complete the project is 1000 man-days, then the time perceived still remaining is 1000/10 = 100 days. If the time elapsed is 50 days, then the indicated completion date is 100+50 = 150 days. If the initial target duration is 130 days and the adjustment delay is 5 days, then the schedule adjustment rate at the first instance is 4 adjusted-days/day. The behavior pattern of schedule adjustment is shown in Figure 3.30. The solid line represents the schedule adjustment goal from an initial value of 130 days to a final value of 150 days. Due to the adjustment time delay, the value of target duration level (represented by dotted line) rises exponentially trying to catch up with the goal. Subsequently, on day 60, the level has achieved 91% of the goal, and by the end of day 70, the level has achieved 99% of the goal.

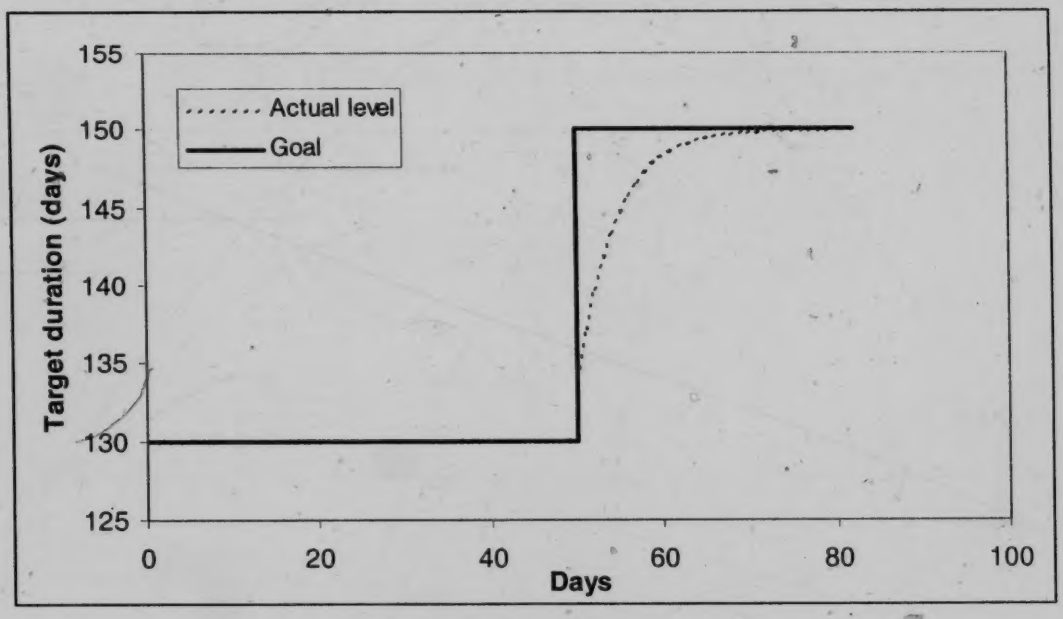


Figure 3.30: Behavior of schedule adjustment

3.3 Scenario Modeling

Software project management using scenario modeling is based on the risk management technique to explore potential causes and outcomes that can affect the project development. The technique is an extension of the system dynamics models incorporating different scenarios and various promising resolution strategies to eliminate and/or reduce the impact of the problems over the project. As previously shown in Figure 1.1, scenario models interact with both shared and individual project system dynamics models. Some of parameters in the system dynamics models are being manipulated in the scenario models to study various potential scenarios. Hence, scenario based project management intends to define a new paradigm for project management (Barros 2000).

As described before in Section 2.4, scenarios represent events, policies, theories, and strategies that are practices that may impose on the project. In order to formulate different scenarios, project managers first need to define several project features or elements that are tied to the above-mentioned scenarios. For example, in the Human Resource model, the "number of employees possibly hired per amount of profits" parameter is a project element related to the scenario policies. This element is a top management constraint imposed on a project. As for scenario strategy, for example, the project manager decides to increase the average daily manpower per staff to cope with the schedule slippage, and the scenario modeling will study the effect of this decision strategy on the project development.

The second step in scenario modeling is to define the range of the project element that is suitable and plausible in the organization. For example, the "number of employees possibly hired per amount of profits" may range from 0 to 10 software developers, i.e., if the organization currently has 100 developers and based on the organization historical data, no more than 10% of the workforce had been hired within a month, then any number greater than 10 is unreasonable in this scenario. Setting a rational range is crucial in the modeling, as otherwise the modeling may generate additional

unnecessary outcomes that may complicate the project manager's understanding of the project behavior.

The next move is to implement the scenarios on the computer generated simulation models using the combination of multi-project network, system dynamics models, and scenario cases, which will be further explained in the Section 4.0.

Finally, when the simulation has generated the outcomes, the project manager needs to perform analysis on the results to understand the underlying relationships. The manager can analyze the simulate events, theories, polices and strategies along with their impact on the project. The scenario modeling may force the manager to think extensively the hidden defining relationships that help compose the software development process. Hence, it is an effective tool to transfer knowledge about process relationships to less experienced project managers. For instance, to study the effect of Brooks' Law on the project schedule, a scenario model can be created by generating a slightly modified graph of WCWF-1 in the Planning model, i.e., allowing the development team to take in new employees in a late project. The result may reveal a further delay in the already late project. The project manager may benefit from this simulation. By analyzing the relationships, the manager may realize that adding manpower later in the project development phase would decrease the productivity as more effort needs to be spent to train and coordinate with the new workers, and thus further extend the project completion date.

3.3.1 Event Scenario

These scenarios may involve some project events that may change the project behavior, and thus affecting some project attributes, like the completion date and workforce level. Event scenarios are associated to specific project elements as generally summarized in the following table.

Element	Model	Scenario Descriptions	Major Potential Impact
Requirement volatility	Effort model	The requirement may change during the project	• Effort needed and workload

		development due to customer's new requests, inaccurate initial specification or underestimation of product features.	• Staff loss due to workload • Schedule
Project size	Effort model	Project size is different for each project, and can be changed during development due to underestimation of size, or change of requirement.	• Effort needed and workload • Staff loss due to workload • Schedule
Project complexity	Effort model	Project complexity varies for each project based on different functionality, interface, reliability and performance. It can be changed during development due to underestimation of complexity or change of requirement.	• Effort needed and workload • Staff loss due to workload • Schedule
Overall software industry unemployment rate	Human Resource model	The unemployment rate varies based on the national and world economies. It can be categorized into low, medium, or high unemployment rate.	• Workers' tendency of quitting • Hiring rate • Total workforce level
Number of projects	Human Resource model	The number of contracts obtained may affect the number of projects available in an organization. The financial implications can be studied in this scenario.	• Profits/Losses • Total workforce level • Training • Productivity
Average profit per project	Human Resource model	The efficiency and productivity of the development team may alter the average profit of a project, and its impact can be modeled in this situation.	• Profits/Losses • Hiring/Firing
Average employment time	Human Resource model	Turnover rate may play an important role in the human resource department. The impact can be studied when the average employment time varies from long to short and vice versa.	• Quitting rate • Experienced staff level • Productivity

Project start/end dates	Planning model	The project start/end dates can be varied to understand the workforce allocation and productivity	• Project workforce • Workload and exhaustion • Total workforce level • %Boost in work rate • Productivity
Maximum tolerable project duration	Planning model	Some critical projects cannot tolerate serious schedule slippages, and the management may need to hire more people to prevent that.	• Willingness to change workforce • Schedule • Project workforce • Total workforce level
Software methodology	Productivity model	Employing different software methodologies, like agile methods, rapid application development, etc may affect the development process	• Productivity
Teamwork cohesion	Productivity model	Enhancing team cohesion through pair programming, joint application design, etc may affect the development process	• Productivity
Personnel motivation	Productivity model	Different levels of motivation can be found in the workforce, and its impact on productivity needs to be understood.	• Productivity

Table 3.4: Event scenario

3.3.2 Policy Scenario

Different organizations may impose different management policies, and thus provide opportunities for the project managers to study different scenarios. The policies represent the tactical behavior patterns affecting the project environment. Some of the policy scenarios in these system dynamics models are described in the following table:

Element	Model	Scenario Descriptions	Major Potential Impact
Renegotiation of contract	Workload and Exhaustion model	When the organization is facing with the problem of staff loss due to workload and exhaustion, the	• Tendency of quitting • Total workforce level • Profits/Losses

		management may need to renegotiate the employees' contract in order to prevent further loss of intelligent assets.	
Average HR cost per employee	Human Resource model	The HR costs per employee may fluctuate when there is higher wage increase, higher mixture of senior staffs, higher spending, etc., and the effect on financial implications can be studied.	• Profits • Hiring/Firing • Total workforce level • Training
Experienced staff over new staff ratio	Human Resource model	Different management may have different policy in hiring experienced and new staffs. Hiring more experienced staff may improve productivity but with a higher HR cost, and vice versa.	• Productivity • Total HR costs
Number of employees possibly hired/fired per amt of profits/losses	Human Resource model	The hiring/firing human resource policies can be studied in this scenario. It can reveal the impact on the project development when a high/low number of workers are suddenly hired/fired.	• Hiring/Firing rates • Total workforce level • Productivity • Schedule • Workload and exhaustion • Project workforce
Amt of training possibly given per amt of profits per month	Productivity model	Training given to employees may vary when the financial status of an organization fluctuates. The effect of training can be examined in this scenario.	• Productivity
Ceiling on target workforce	Planning model	Management may impose a ceiling on the number of workers hired in a project, and the change of this parameter may affect certain project behavior.	• Total workforce level • Schedule • Project workforce • Workload and exhaustion

Table 3.5: Policy scenario

3.3.3 Theory Scenario

Theory scenarios represent behavior patterns that the project manager believes may influence the project (Barros 2000). They involve the proven or the hypothetical

management theories obtained from common sense, experience or expert's advice. There is one proven theory that can be modeled in this research, i.e., Brooks' law: "Adding manpower to a late software project makes it later" (Brooks 1995). As described in the Planning model, WCWF-1 is the willingness to change workforce level that is based on the workforce stability. The general form of WCWF-1 graph (Figure 3.28) states that the management is less willing to take in new workers when there is less time remained in a project, but is more willing to hire them during the early phase of the development. To test this theory, the graph can be shifted further to the right, in such a way that the graph intercepts the X-axis with a value less than 0.3 as shown in Figure 3.31.

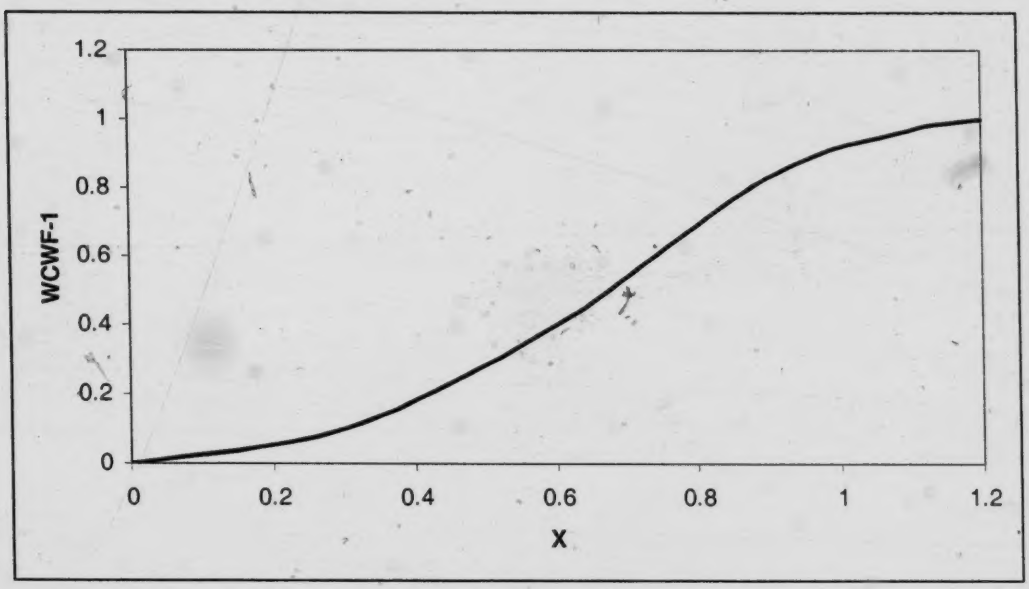


Figure 3.31: Modified form of WCWF-1

$$\text{Where, } X = \frac{Time\ remaining}{(Hiring\ delay + Assimilation\ delay)}$$

The illustration demonstrates that when the time remaining in a project drops below 0.3 x (Hiring Delay + Assimilation delay), the management is still willing to hire some people into the project. When this graph is used in the Planning model, the effect of Brooks' law can be simulated. The results may show an increase of

workforce during the late development, and as a result of new employees yet to go through the assimilation process, the productivity may drop and thus further delay the project.

3.3.4 Strategy Scenario

These scenarios represent the decisions and action plans the manager may execute during the project development. These strategies may have short-term or long-term effects on the project. The following table describes some of the scenarios found in this research:

Element	Model	Scenario Descriptions	Major Potential Impact
Nominal fraction of a man-day on project	Workload and Exhaustion model	The nominal fraction of a man-day on a project states the portion of the day working on the project. Hence, when the management has decided to change the value by increasing/decreasing it, the effect can be seen in the model.	• Exhaustion • Ability to cope • Total workforce level
Assimilation delay	Human Resource model	The management can reduce/increase the assimilation period of the new workers by providing more/less onsite training, or hire more/less experienced new workers instead of fresh graduates.	• Experienced staff's level • Productivity • Schedule • Project workforce
New/Experienced staff's nominal productivity	Productivity model	From time to time, the nominal productivity may change due to better/worse management control, and better/worse vertical communication level with the top management.	• Productivity
Average daily manpower per staff	Planning model	This value can be varied to see the effect of spending more/less time per day on the project. In addition, it is an indication of multi-tasking. This scenario models the management decision of the	• Productivity • Project workforce • Schedule

		number of projects a worker can multi-task at one time.	
Project end date	Planning model	The manager may create a false schedule pressure for the development team by altering the project end date. The false schedule pressure is created when the workers perceive the project to be behind schedule. This management strategy may improve the productivity.	• Productivity • Workload and exhaustion • Schedule • Project workforce

Table 3.6: Strategy scenario

3.3.5 Application of Scenario Modeling

To emphasize again the purpose of scenario planning, its intention is not to predict the future, but to provide an understanding of some of the plausible and possible futures. Hence, there could be an infinite amount of possible futures that we can study, but our focus is to study those that matter by identifying the key driving forces or elements. Those elements represent the uncertainties that may lead to different scenarios. At first, the amount of uncertainties may seem overwhelmingly large, as our real world is full of infinite amount of uncertain variables. Nevertheless, by stepping back, we can reduce the bundles of uncertainties by looking at the commonality of uncertainties with a single spectrum, using an axis. If we simplify our entire list of key uncertainties into only two orthogonal axes, then we can define a matrix that has four different, but plausible, quadrants of futures (Sherwood 2002) (Punie 2001). Each of these four corners is a logical future that we can explore, as shown in the following diagram.

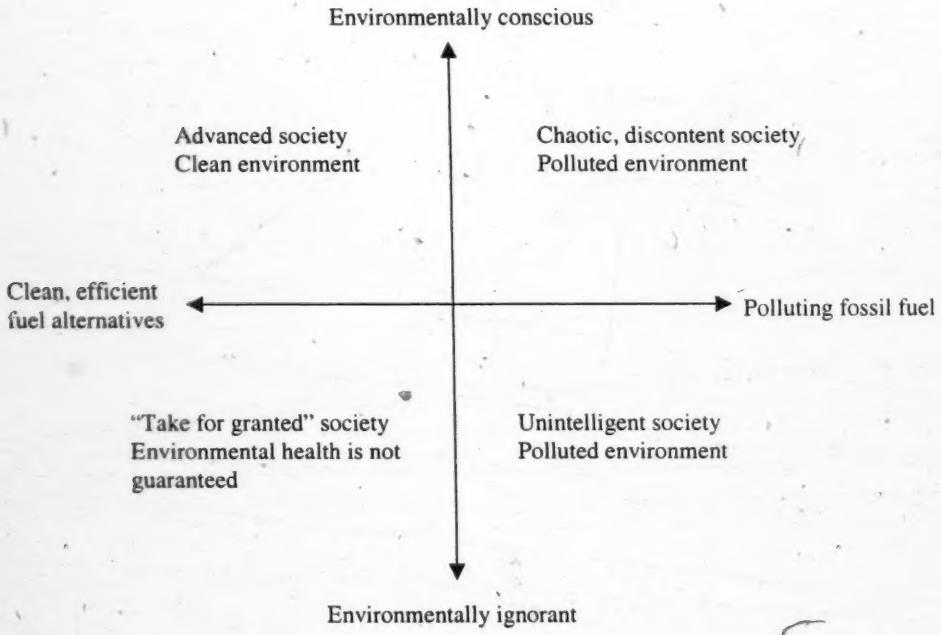


Figure 3.32: Example of scenario quadrants

The diagram is self-explanatory, and it portrays significantly different paths in the future with two key uncertainties, i.e., environment consciousness and fuel technology. Those four quadrants are the results of the combination of these two uncertainties. Of course, we can generate more scenario quadrants by using more than two axes, for example, 3-dimensional axes with three uncertainties can generate eight possible futures. However, fewer are better as scenario planning is not about spawning an entire tree of possible paths, but an effective and clear way for management to make decisions.

Upon examining these four scenario types and scenario quadrants, some insights are provided to illustrate the application using the project elements found in Sections 3.3.1-3.3.4.

3.3.5.1 Event Element vs. Event Element

During the project development, if there are two possible events that may occur, such as requirement change and tolerable project duration, the interaction of these two

event elements may create different outcomes. Requirement volatility is considered high when the requirements are constantly altered and redefined, may be due to customer's needs or inaccuracy in defining specifications. The maximum tolerable project duration is high when a project can be tolerated to be late, and is low when a project has to be completed on time. The following diagram shows the possible four outcomes based on these two uncertain event elements.

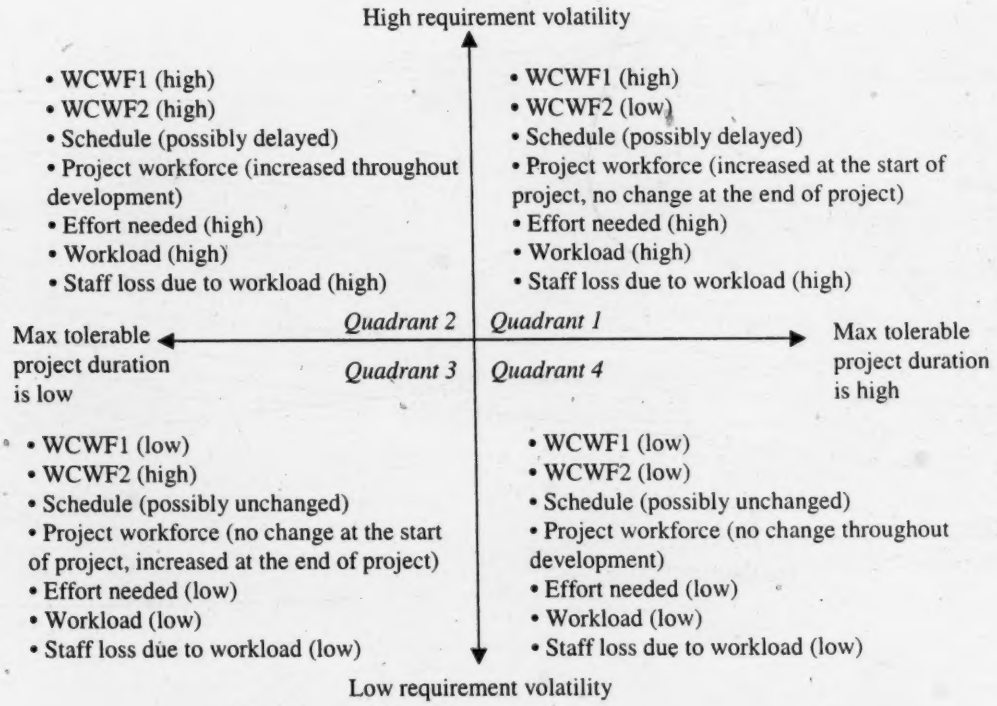

High requirement volatility
• WCWF1 (high)
• WCWF2 (high)
• Schedule (possibly delayed)
• Project workforce (increased throughout development)
• Effort needed (high)
• Workload (high)
• Staff loss due to workload (high)
• WCWF1 (high)
• WCWF2 (low)
• Schedule (possibly delayed)
• Project workforce (increased at the start of project, no change at the end of project)
• Effort needed (high)
• Workload (high)
• Staff loss due to workload (high)
Quadrant 2
Quadrant 1
Max tolerable project duration is low
Max tolerable project duration is high
Quadrant 3
Quadrant 4
• WCWF1 (low)
• WCWF2 (high)
• Schedule (possibly unchanged)
• Project workforce (no change at the start of project, increased at the end of project)
• Effort needed (low)
• Workload (low)
• Staff loss due to workload (low)
• WCWF1 (low)
• WCWF2 (low)
• Schedule (possibly unchanged)
• Project workforce (no change throughout development)
• Effort needed (low)
• Workload (low)
• Staff loss due to workload (low)
Low requirement volatility

Figure 3.33: Requirement volatility vs. Maximum tolerable project duration

The sectors are numbered for ease of illustration. In Quadrant 1, when a project has high volatility in changing requirements and the project can be tolerated to be late, it is very likely that the target schedule is going to be delayed. The willingness to change workforce due to workforce stability (WCWF1) is also high. The reason is since the schedule can be delayed, the time remaining can be increased. Thus, the time remaining may be much larger than the total delay time required to hire new people and to train them during assimilation period. As a result, based on Brooks' law, the management is more willing to change the workforce level when the

remaining development time is perceived to be sufficient to hire and train new workers. On the contrary, the willingness to change workforce due to schedule stability (WCWF2) is low. As WCWF2 is a function of target duration over maximum tolerable project duration, a project has high flexibility in delivery time, and so the management is less willing to increase the workforce later in the project, as adding more people late in a project may further delay it. Due to the nature of high WCWF1 and low WCWF2, the project workforce is likely to be increased at the start of the project, and the workforce level remains stable when the project development approaches the deadline. As the requirements are always changing, the effort needed to handle the project is increased due to unexpected changes and corrections. Consequently, the workload is high and may result in loss of staff due to workload.

In Quadrant 2, the project suffers high requirement changes and the management has a low tolerance for late project. Hence, the willingness to change workforce due to both workforce and schedule stability is high. As the management anticipates many requirement adjustments, the project workforce is increased throughout the project development. Consequently, it is likely the final project schedule is delayed as more new workforce is added at the end of the project and results in a reduction in productivity due to time spent in training.

In Quadrant 3, the project is expected to be completed on time with not much tolerance in delay, but the project requirements are not expected to change vastly throughout the development. As a result, if proper planning is done at the start of the project, it is likely the project will be completed on schedule. Hence, as the perceived time remaining remains stable, WCWF1 is low or possibly unaffected at the start of the project. Nevertheless, WCWF2 can be high if due to poor planning and the workforce has to be increased at the end of the project to ensure on time delivery. The effort needed and workload are considered low due to the stable requirements.

In Quadrant 4, the project is low in requirement changes and the management can allow an extension to the final schedule. Hence, the willingness to change the

workforce due to workforce and schedule stability is considered low. This project is the most stable of all quadrants and receive the least attention from the management due to the flexibility in target duration. So, with proper planning, the final schedule is likely unaffected and the project workforce stays stable throughout the development. The effort needed is the lowest of all, along with the workload. Consequently, the staff loss due to workload is minimum.

3.3.5.2 Policy Element vs. Event Element

One of the policy elements stated in Section 3.3.2 is the average human resource cost per employee. This is directly related to the organizational spending policy on human resource, for example, the policy may allow unrestricted departmental expenditure on travel expenses, and/or may allow high salary increment among workers, etc. The number of projects represents the available income in an organization. In our scenario, we assume the average profit per project is constant for all quadrants. The interaction between these two elements generates the following diagram.

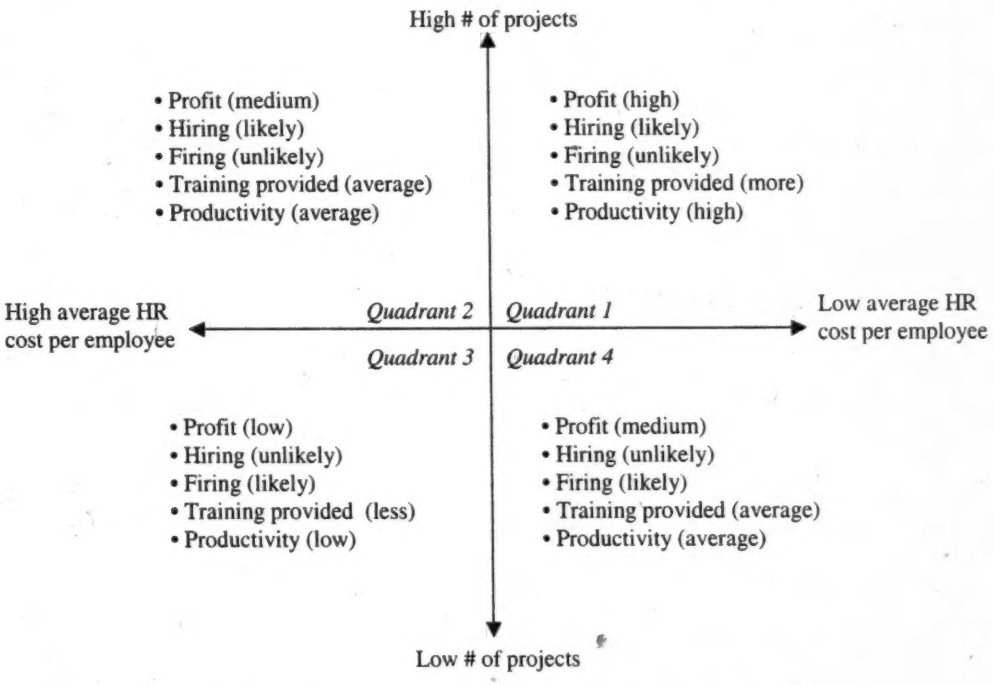


Figure 3.34: Average HR cost per employee vs. Number of projects

In Quadrant 1, this is the most efficient organization, by obtaining a large quantity of projects while maintaining a low average HR cost. As a result, the generated profits are high. Due to a high volume in projects, the organization is willing to hire more workers when needed, and is more unlikely to fire its workers. As the amount of training given to its employees is believed to be related to the available profits, the workers may receive more training than other quadrants, and consequently, they experience an increase in productivity.

In the second quadrant, the organization spends more money on the HR cost, and despite a high number of projects developed, the generated profits are less compared to the first quadrant. Nevertheless, the company is still willing to hire workers to handle the large volume of projects, and is less willing to fire its workers, which in return increases the HR cost. Due to the mediocre profits, the amount of training available is average, and so is the productivity.

In Quadrant 3, this is the worst scenario for an organization when there is a limited amount of projects in hand while the money spent on human resource remains high. Hence, the generated profits are low, and new workers are less likely to be hired due to low number of projects, while existing workers are more likely to be fired. The amount of training is comparatively the lowest of all quadrants, along with the productivity.

In the last quadrant, the company understands the need to conserve financial resources when the business is declining. The HR costs remain low, and in order to achieve that, it is likely some workers will be fired while freezing the hiring process. Due to this conservative policy, the profits remain average, along with the amount of training and productivity.

3.3.5.3 Strategy Element vs. Policy Element

In terms of strategic management, a manager may create a false schedule pressure for the development team by reducing the project end date. The sense of schedule

pressure is created when the developers perceive the project to be behind schedule. Conversely, the manager may reduce this pressure by increasing the perceived project end date, although this is rarely practiced in the real world. The ceiling on target workforce is the maximum limit of workforce in a project development team. The ceiling can be increased so that more workers can be hired, or can be reduced so that a minimum number of workers is retained. The interaction of these two elements is studied in the following diagram.

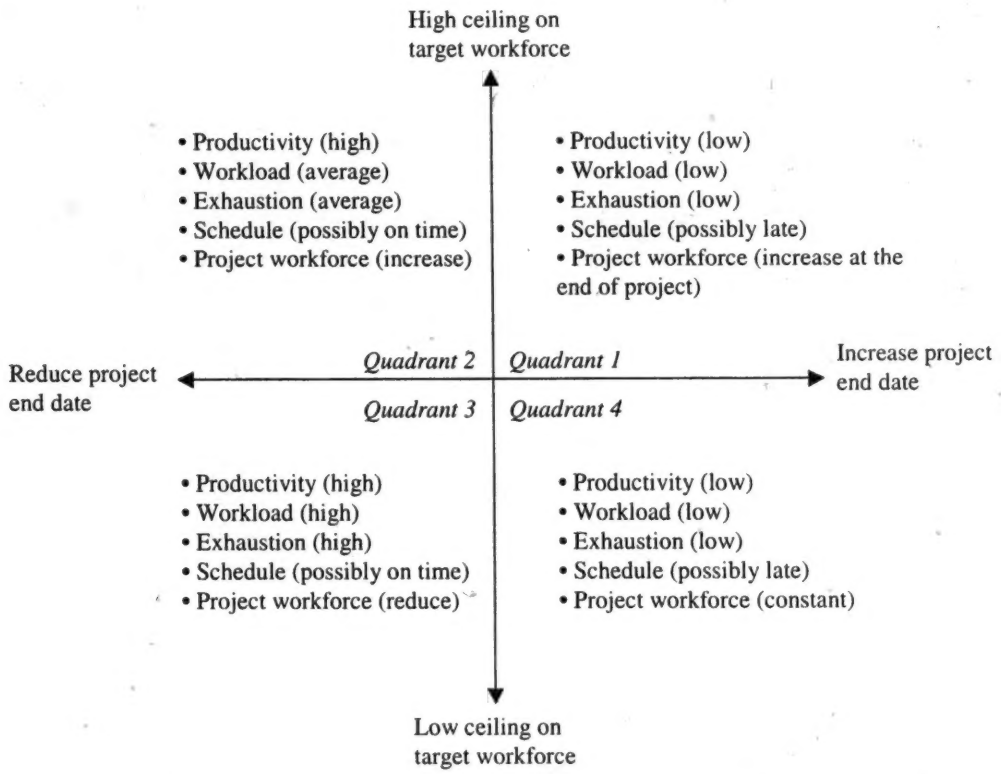


Figure 3.35: Project end date vs. Ceiling on target workforce

In the first quadrant, the perceived project end date is greater than the actual end date, and the development team can hire more workers if needed with the high ceiling. Consequently, since the team falsely believes that they have sufficient time to complete the project, the fraction of a man-day on the project is spent less than the nominal, and thus resulting in a low productivity. Similarly, with an extended duration, the workload is less, along with minimum exhaustion. Due to the lack of

urgency, the project may possibly be behind schedule. When the team realizes that the actual due date is sooner than they perceived, they may need to increase the workforce later in the development to compensate for the loss time, but as stated by the Brooks' law, this action still results in late schedule.

In Quadrant 2, the manager purposely reduces the perceived end date but the development team is allowed to hire more workers. As a result of the shortened duration, the team needs to spend more time working on the project, and thus an increase in productivity. If the need arises, they may hire more workers so that their workload and exhaustion are kept at bay without overworking the existing workforce. Due to this combination of outcomes, the final schedule may not be jeopardized, and if more workers are hired in the process, the workforce level is increased.

In the next quadrant, the manager creates the same false increased schedule pressure, and the organizational policy just happens to be limiting the intake of workers with a low ceiling. With the same reason as the previous quadrant, the productivity is enhanced by increasing the fraction of a man-day spent on the project. However, as the team cannot hire many new workers, the workload felt by the existing workforce increases due to the shortened timeline and limited resources. If the exhaustion persists throughout a long duration, some workers may leave the organization due to inability to cope with their jobs. Nevertheless, as long as the staff loss is insignificant, the project is still likely to be completed on time as the productivity has been increased.

In the fourth quadrant, the perceived end date is longer than the actual target date, and the team has low ceiling on workforce. As the project duration is believed to be extended, the development team may observe a reduction in productivity. Similarly, the workload is low with minimum exhaustion. Due to the false sense of time perception, the project is possibly delayed. As there is a low ceiling on target workforce, the project team size may remain constant throughout the development.

3.3.5.4 Theory Element vs. Policy Element

The theory scenario, discussed in Section 3.3.3, is related to Brooks' law. There are other theories in software engineering management, but only one theory can be applied in the system dynamics models outlined in this research, as the fundamental causal effect relationships that imitate a theory need to exist in the models before the theory can be tested. Hence, a similar theory is used in the following example, along with a policy element, i.e., the experienced staff over the new staff ratio. An organization may decide on the ratio based on various criteria, such as the availability of experienced staffs, the HR cost, the productivity, the need for experienced staffs, and etc.

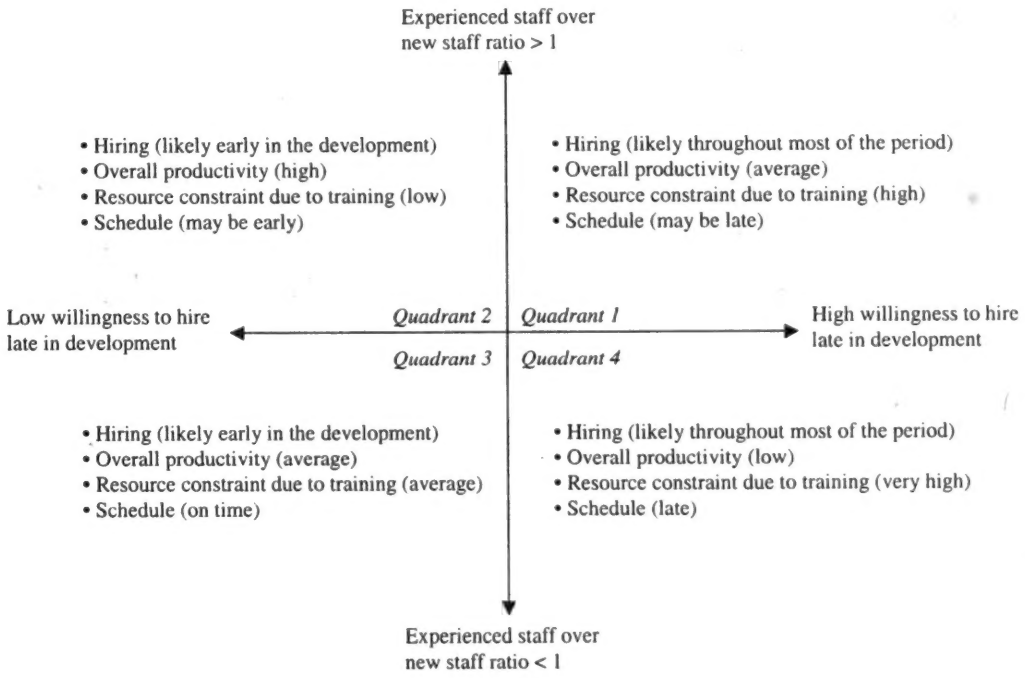

Experienced staff over new staff ratio > 1
• Hiring (likely early in the development)
• Overall productivity (high)
• Resource constraint due to training (low)
• Schedule (may be early)
• Hiring (likely throughout most of the period)
• Overall productivity (average)
• Resource constraint due to training (high)
• Schedule (may be late)
Low willingness to hire late in development
Quadrant 2
Quadrant 1
High willingness to hire late in development
Quadrant 3
Quadrant 4
• Hiring (likely early in the development)
• Overall productivity (average)
• Resource constraint due to training (average)
• Schedule (on time)
• Hiring (likely throughout most of the period)
• Overall productivity (low)
• Resource constraint due to training (very high)
• Schedule (late)
Experienced staff over new staff ratio < 1

Figure 3.36: Willingness to hire late in development vs. Experienced staff over new staff ratio

In Quadrant 1, the organizational policy is to hire more experienced staffs over new staffs, and when the theory of Brooks' law is not followed in the project development, the following situations may occur. When the hiring continues throughout almost the entire development, despite a higher percentage of experienced staffs in the project

team, the overall productivity may drop from high to average. Some of the experienced staffs may have to allocate certain amount of time and effort to train the new workers, even when the project approaches the deadline. Hence, the strain on the human resources is comparatively high and the project may suffer delay if the resources are not well managed.

In the second quadrant, the same policy is applied but the Brooks' law is complied in the organization. As a result, the overall productivity is not affected by any late hiring, and since there are more experienced staffs than new staffs, the productivity is consistently high throughout the development. The resource constraint due to spending time training the new workers is relatively low, as the training is only conducted early in the development. Suffice to say, the project may be finished on time, and in some cases, with prime efficiency, it may be completed sooner.

In the third quadrant, the organization may have a different policy in hiring workers, emphasizing more on inexperienced employees than skilled employees. However, the hiring is only limited to the early phase of development. The new workers may have enough time to gradually build their skills and experience, and increase their productivity from low to average. As there are less experienced staffs in this quadrant than the previous quadrant to perform the training, the constraint on resources is relatively higher than the second quadrant's. Overall, with an average productivity, the project is likely to be completed on time.

In the last quadrant, the organization prefers or is forced to hire more new workers than experienced workers. If the hiring continues throughout the development, the overall productivity may drop from average to the lowest of all quadrants. The main reason is with a smaller percentage of skilled workers on board, they are tied with training the new workers throughout the development as well as developing the project. Hence, the overall productivity is relatively low with a very high constraint on resources. As a result, this quadrant has the highest possibility to finish the project late.

While the above four scenario examples express some representative scenarios, it is by no means the only scenarios available. Different new scenarios can be created by combining and altering different elements through "mix and match". For example, by creating a false schedule (as in Strategy scenario) and shortening the maximum tolerable project duration (as in Event scenario) concurrently, the development team may experience a higher schedule pressure than before, and the effect may result in working overtime in order to meet the target dateline. Upon examining these scenario elements, different outcomes and conclusions can be drawn from the simulation with different configurations. Hence, the scenario modeling is a good means to test different events, policies, theories and strategies before implementing it during the project development.

4.0 MODEL INTEGRATION

After describing the principles and functionalities of the various models, the integration of these models to create a unified and functional multi-project management system is illustrated in the following diagram. There are 5 basic components as shown in Figure 4.1:

- Scenario model
- Multi-project network model
- Controller
- Shared models
- Individual project models

All the models have been discussed in the previous sections, however, the discussions are only limited to their own domains, without the consideration of integrating with another models. The integration of various models may bring forth a range of issues, like parameter passing with another models, command flow, internal dynamics, and controllability. This section describes the integration method using a central controller to iron out those issues, and is then followed by two examples to illustrate the process.

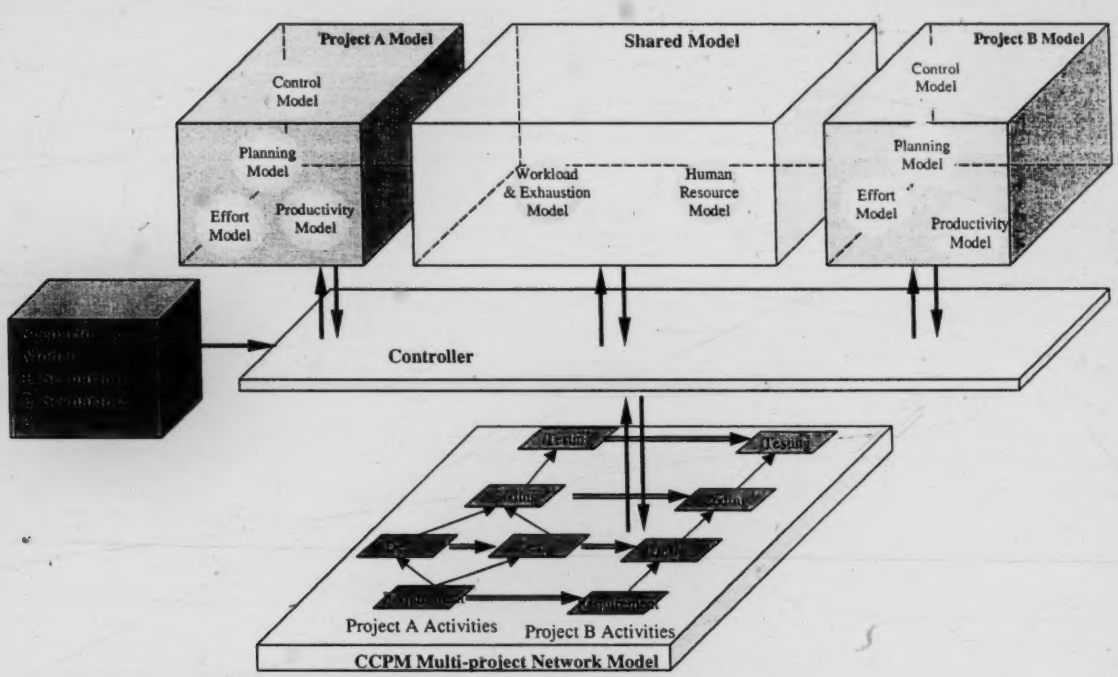


Figure 4.1: A Simplified View of Multi-Project Management Model

The controller is the center component that controls the system dynamics models and the multi-project network. Using the scenarios created in the Scenario model, it will pass the parameters or elements to the controller. For example, the project size and complexity are varied by the managers in different scenarios and then are delivered to the controller. Hence, the Scenario model also acts as a user interface with the system. Based on the scenario requirements, the controller will coordinate the information flow between the system dynamics models and the Scenario model. As an example, when the element "overall software industry unemployment rate" is supplied to the controller, it will channel the information to the Shared model in the Human Resource model, and for any individual project related parameters, like project size, complexity and etc., they are sent to its individual project model.

The controller may also communicate with the CCPM network. The network is supplied by the management to layout its multi-project environment with project start and end dates. The controller will receive this duration information from CCPM, and supply to its respective project in the Planning model. As described before, the

Planning model will then determine the required project workforce and relay the information back to the controller. The controller will then draw the appropriate resources from the shared model, and communicate back to the CCPM network about the respective resource allocation.

The information sharing between the individual and shared models will take place in the controller. For example, the number of tasks in the Effort model and the time remaining in the Planning model will be conveyed to the Workload and Exhaustion shared model via the controller, and vice versa.

It will also interact with the individual project models to simulate certain activities, like Control or Planning that exist within the respective projects. The results of the simulation will be sent back to the controller for further action. If certain variations, like indicated completion date and schedule adjustment, in the dynamic model exist and cause the delay of a project, the controller will feedback the change to the multi-project network. If a critical chain in the project is affected by the delay, the CCPM network will then have to reallocate the resource based on the latest information. The outcome of this readjustment may result in the delay of another project completion date, and rearrangement of the resource allocation. In return, the affected project will have to start late, and the schedule pressure and productivity within the individual project model will be affected. Consequently, the effect of multi-project dependencies is portrayed in the multi-project management model through the integration of these four types of models and a controller.

4.1 EXAMPLES

To illustrate the working and the application of this proposed system modeling, we employ the following examples. Using the same projects as in Section 3.1 with the same criteria, we want to study the impact of increased project size on the project duration, manpower requirements, and exhaustion due to overwork. The scenarios are stated as follow:

Scenario 1: The requirement size (measured in function points) in Activity A2 is increased by 40%, but the number of staff and productivity are not allowed to be increased. Hence, the project completion date has to be delayed and we want to study the extent of the Project I and II durations affected by the increased workload.

Scenario 2: The requirement size (measured in function points) in Activity A2 is increased by 40%, but the project duration is not allowed to be changed, and it is assumed that the employees do not work overtime. Hence, we want to study the number of staff affected by the increased workload performed in a fixed amount of time. To simplify the case, we only study the number of increased staff for Resource A in activity A2, although all resources will be affected in this scenario.

Scenario 3: The requirement size (measured in function points) in Activity A2 is increased by 40% after the 4^{th} week of development. The project duration does not change as in Scenario 2, but the organization does not want to hire new employees for the increased workload. Their only resort is to force the employees to work overtime and increase their productivity to compensate for the workload and fixed schedule. Hence, we want to study if this strategy is feasible, and the impact of workload and exhaustion on the employees.

Now, referring to Table 4.1, we assume the following project properties for all scenarios.

#	Attribute	Value
1	Potential productivity for resource A	2.5 function points/man-week
2	Total available manpower of resource A	12 engineers
3	Communication overhead (Com)	Com = 0.0006 * (manpwr)^2 manpwr is the workforce sought
4	Original size of Project II	500 function points
5	New size of Project II	700 function points

Table 4.1: Project Properties

4.1.1 Scenario 1 Modeling

Using the Critical Chain Project Management in Section 3.1, the following figure (Figure 4.2) shows the durations of each activity before the increment of the project size. Figure 4.3 is an equivalent diagram in Gantt chart representation. Early start (ES), and early finish (EF) for each activity are first determined using a forward pass, and then the late finish (LF) and late start (LS) are calculated using a backward pass. The slack is the difference between the late start and the early start.

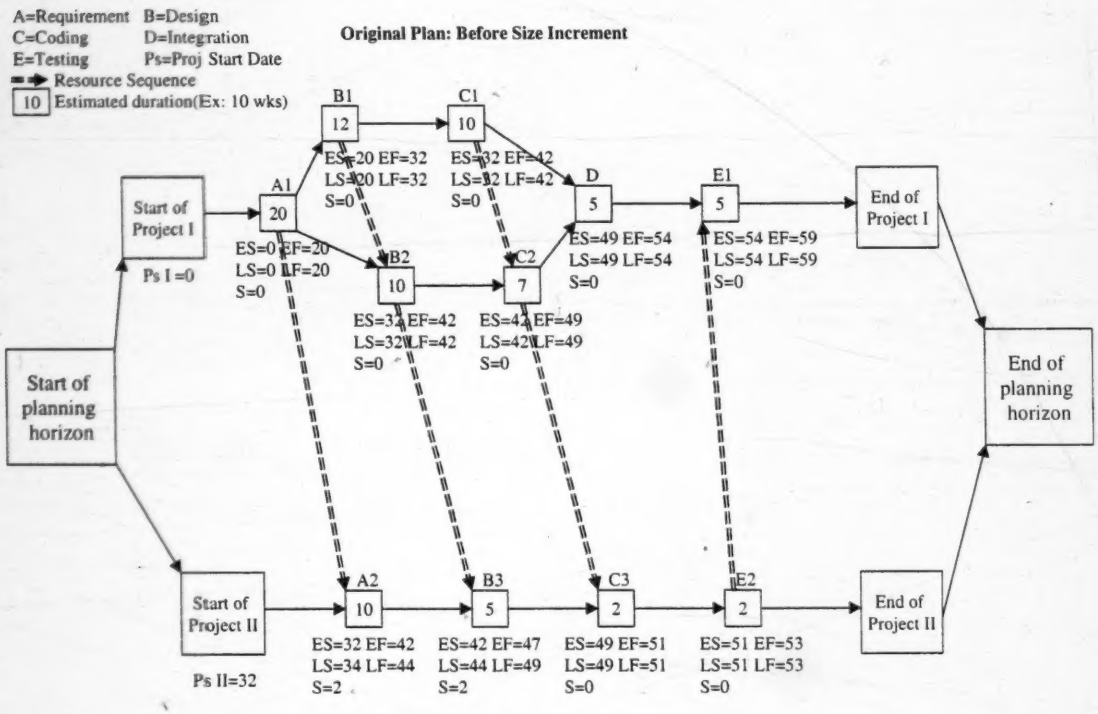


Figure 4.2: CCPM network before size increment

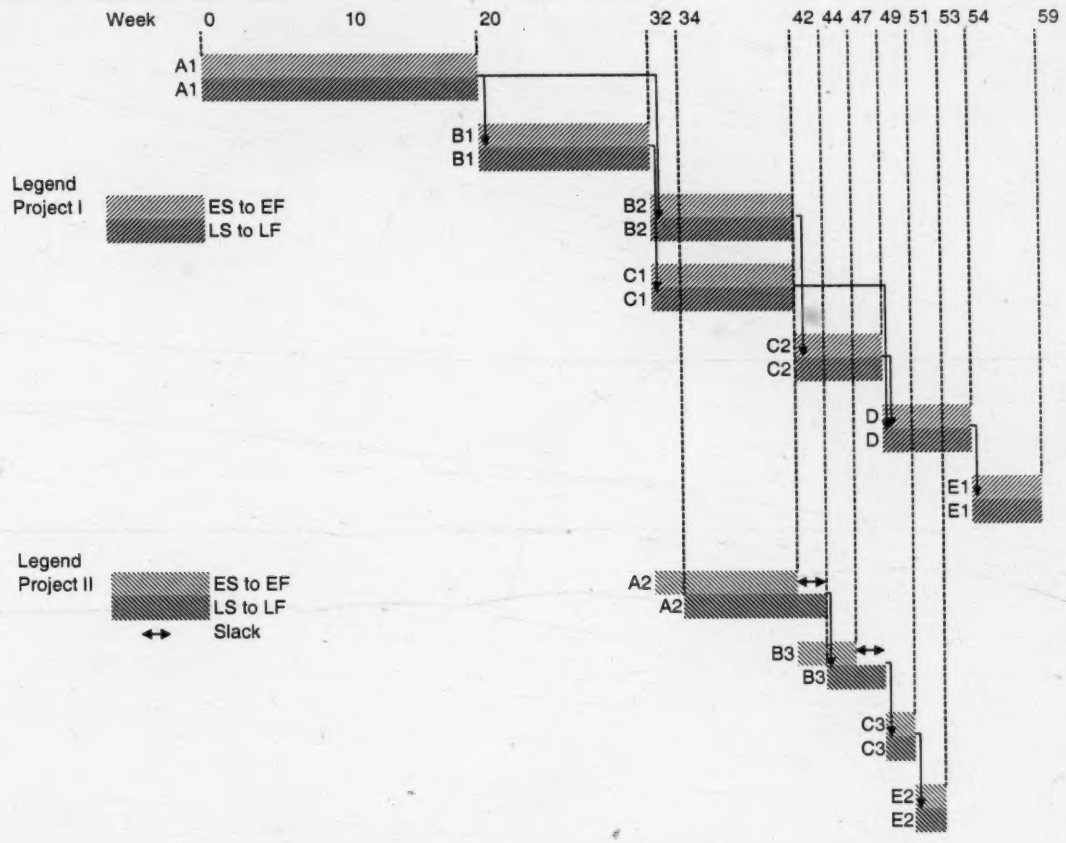


Figure 4.3: Gantt chart representation before size increment

When Scenario 1 is created by a project manager, the Scenario model will send the size increment to the Controller as shown in Figure 4.4. The Controller, using a built-in function, will call the Multi-project Network model and convey the respective parameters, i.e., project affected and size increment, to the model. The network model will recalculate the durations for each activity and reschedule the resource allocation (Figure 4.5). Upon completion, the expected project duration for each project will be returned to the Controller, and it will then update the system dynamics models with the new durations.

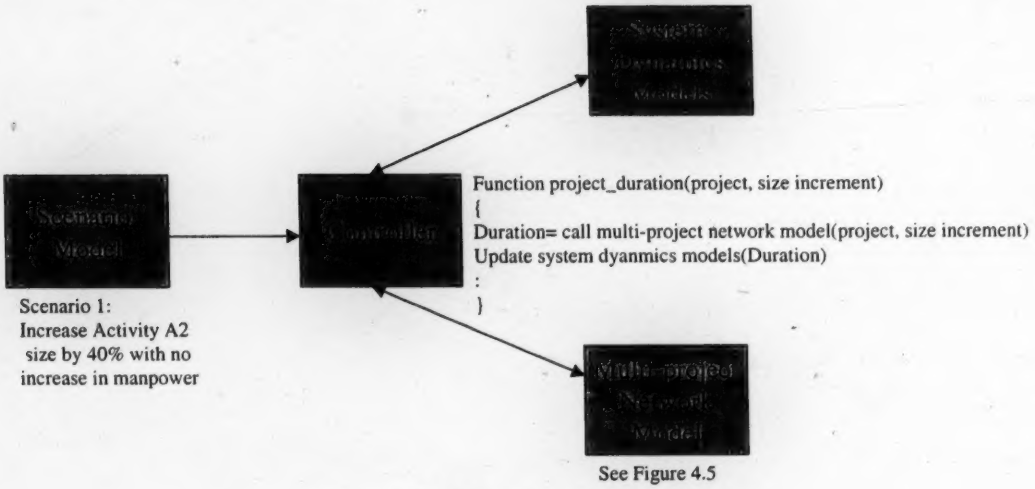


Figure 4.4: Scenario 1 Modeling

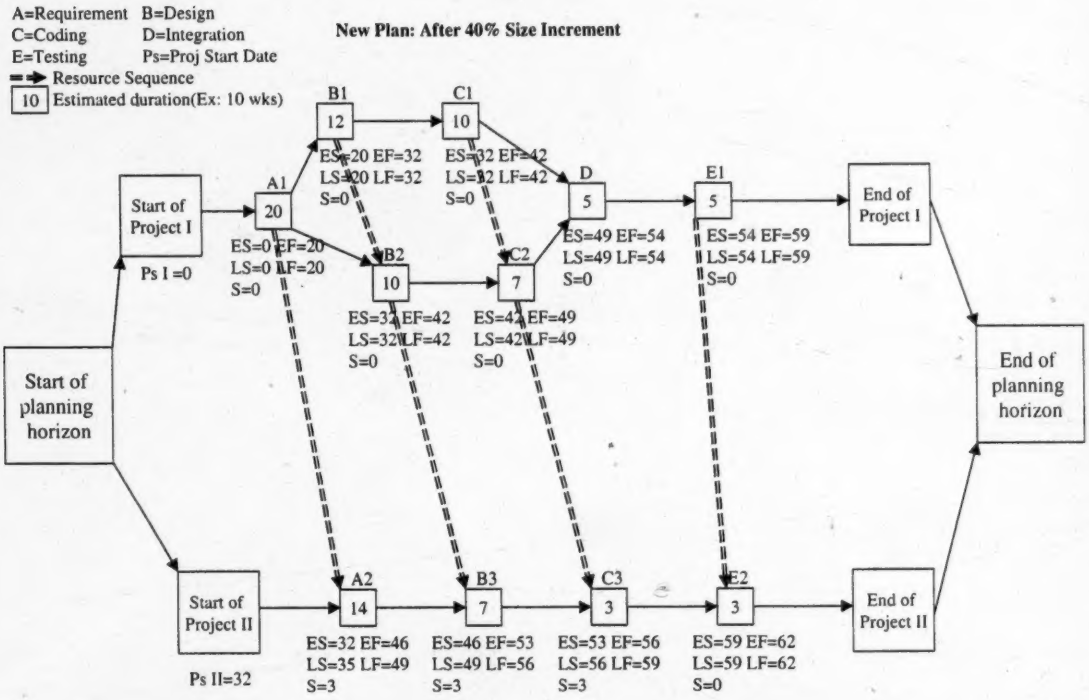


Figure 4.5: CCPM network after 40% size increment in Project II

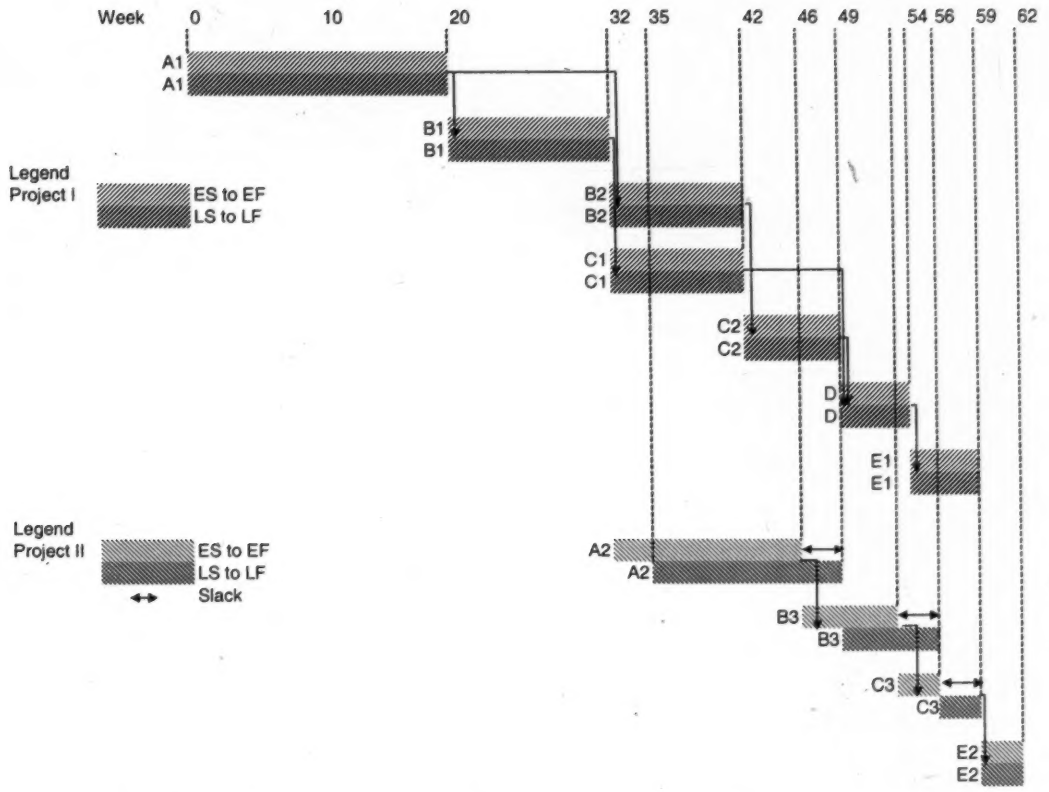


Figure 4.6: Gantt chart representation after 40% size increment in Project II

In Figure 4.5 and Figure 4.6, as the requirement size has been increased by 40%, the durations for the subsequent activities are increased by a proportional amount. As more function points are required to be built into the project, the activities involving design, coding and testing will need to expand to cope with the larger project size. Hence, all the activity durations in Project II have been increased by 40% each. Consequently, the overall Project II duration has been increased by 43% and the duration of Project I is not affected. This is because when allocating Resource E, the Project I has higher priority than Project II. So, the workflow of Resource E has been changed from the original plan, and the Resource E will perform Project I first without waiting for Project II to complete. After finishing activity C3 in Project II, the project is delayed for 3 weeks before Resource E is transferred from Project I to Project II. Hence, by using the model, we can determine the impact of size increment

on each project and observe the changes made to the multi-project network plan. The following table (Table 4.2) summarizes the results for Scenario 1.

#	Original Plan (No increment)	New Plan (40% increment)
1	End of Project I at 59 weeks	End of Project I at 59 weeks
2	End of Project II at 53 weeks Total project time: 21 weeks	End of Project II at 62 weeks Total project time: 30 weeks
3	Critical chain for Project II: A1-B1-B2-C2-C3-E2	Critical chain for Project II: A1-B1/B2-C1/C2-D-E1-E2
4	Resource E workflow: E2 -> E1	Resource E workflow: E1 -> E2

Table 4.2: Scenario 1 Results

4.1.2 Scenario 2 Modeling

In Scenario 2, the project duration cannot be altered when the size of the project is increased, and the employees do not have to work overtime or increase their productivity to meet the deadline. Hence, the organization resolves this issue by hiring new employees, and we want to estimate the number of additional workers to be hired. As shown in Figure 4.7, the controller will call the system dynamics model to compute the manpower sought for Resource A with the specified project and size increment. As the duration of the projects is not being affected, the CCPM network is not involved in this scenario. The detailed resource computations of the model are shown in Figure 4.8. The displayed model is a simplified combined version of Effort model, Productivity model, Planning model and Human Resource model. A simplified version is used, as the main purpose of the example is to illustrate the working of the entire multi-project management modeling as a whole, and not its individual models.

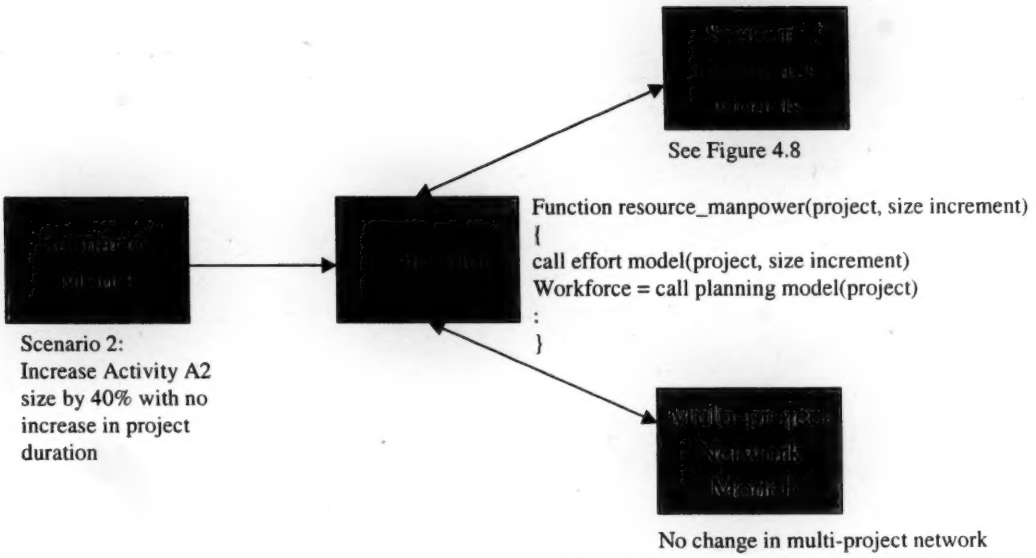


Figure 4.7: Scenario 2 Modeling

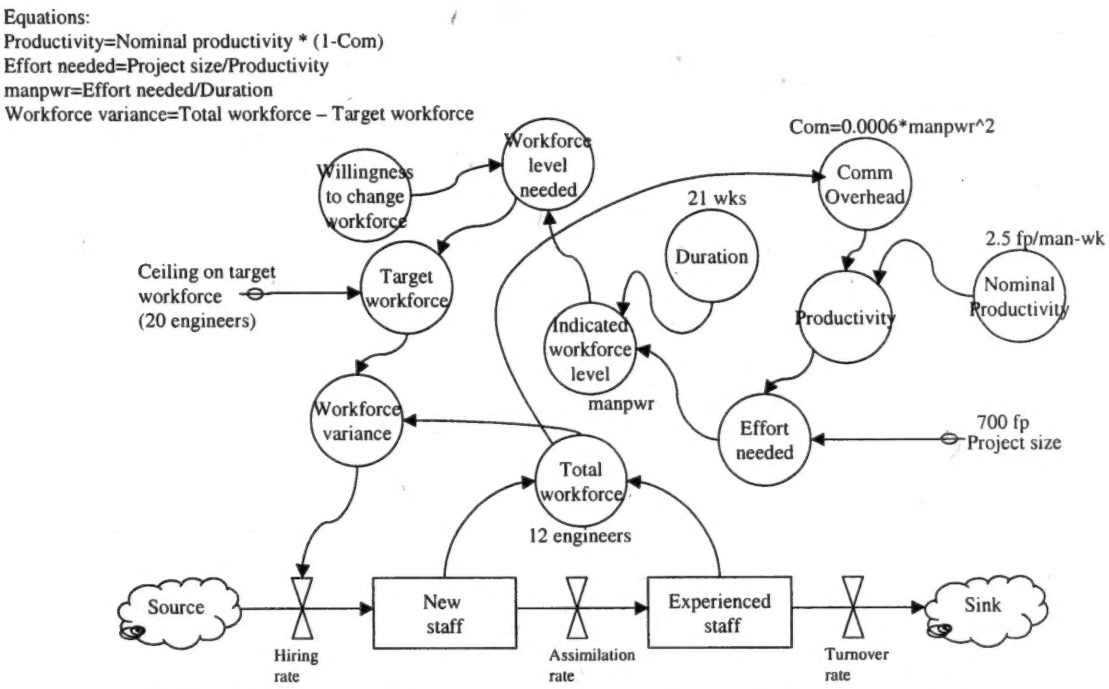


Figure 4.8: Combined system dynamics model with 40% size increment

Figure 4.8 shows the equations and the relationships involved in the combined model, needed to compute the target workforce and the workforce variance. The willingness to change workforce is considered 1, as the project is in the initial stage of development, and hence, the workforce level needed is identical to the indicated workforce level. The target workforce is compared with the ceiling on target workforce, and since the needed level is less than the value of the ceiling, the target workforce equals the workforce level needed. The variance is the difference between the total workforce and the target workforce.

The following table (Table 4.3) shows the summary of the computations of the model for the original and the new plans:

#	Variable	Original Plan (no increment)	New Plan (40% size increment)
1	Project size	500 function points	700 functions points
2	Productivity	2.32 function points/man-wk	2.12 function points/man-wk
3	Effort needed	215.66 man-wk	330.81 man-wk
4	manpwr	10.3 ≈ 11 engineers	15.8 ≈ 16 engineers
5	Workforce variance	Extra 1 engineer	Lack 4 engineers

Table 4.3: Scenario 2 Results

When the project size increases by 40%, the productivity drops by 9% due to the increased communication overhead. As more communication is needed, the efficiency of the workforce will drop. As a result, the demand of the workforce sought increases from 11 engineers to 16 engineers. As the organization has 12 available engineers, the management needs to hire additional 4 new engineers to meet the deadline of the project. Hence, without compromising the original schedule, the organization can increase their workforce through hiring in order to get the work done on time and without driving their employees working overtime.

4.1.3 Scenario 3 Modeling

In Scenario 3, the management has no tendency to increase the workforce and still wants to meet the original deadline by increasing their employees' productivity and working hours. The question is whether the development team can achieve this goal by following this strategy. This scenario studies the feasibility of this strategy by looking at the employees' overtime, workload, exhaustion, quitting rate, productivity, ability to cope, and the project cumulative tasks developed in the process.

Before analyzing this scenario, the following criteria and assumptions are made as in Table 4.4 and Table 4.5:

#	Attribute	Value
1	Project II original size and expected completion duration	500 function points, 21 weeks
2	Project II new size	700 function points after the 4th week of development
3	Project II team size	12 engineers initially
4	Nominal Productivity	2.284 function points/man-wk
5	Average daily manpower per staff	1 (Full time)
6	Nominal workload	2 function points/man/wk
7	Overwork duration threshold	10 weeks
8	Nominal fraction of a man-day	0.7 (Out of 8hr working day, 5.6hr spent on project)
9	Max boost in working hours	100% (Max spend 1.4 fraction of a man-day on project)
10	Nominal exhaustion level	5
11	Resignation notice	4 weeks

Table 4.4: Scenario 3 Criteria and Assumptions

Tendency of quitting	Ability to cope	Weekly quitting rate %
None	>= 1.0	0%
May be/Slightly	>= -1.0, <1.0	1%
Likely	< -1.0	2%

Table 4.5: Quitting rate due to inability to cope

The modeling was based on the relationship and formula found in Section 3.2, and calculations were done using Microsoft Excel. The results are tabulated in Appendix 1, detailing the workload, ability to cope, cumulative tasks developed, and other relevant data for each week. Some important data are extracted and plotted in the following diagrams.

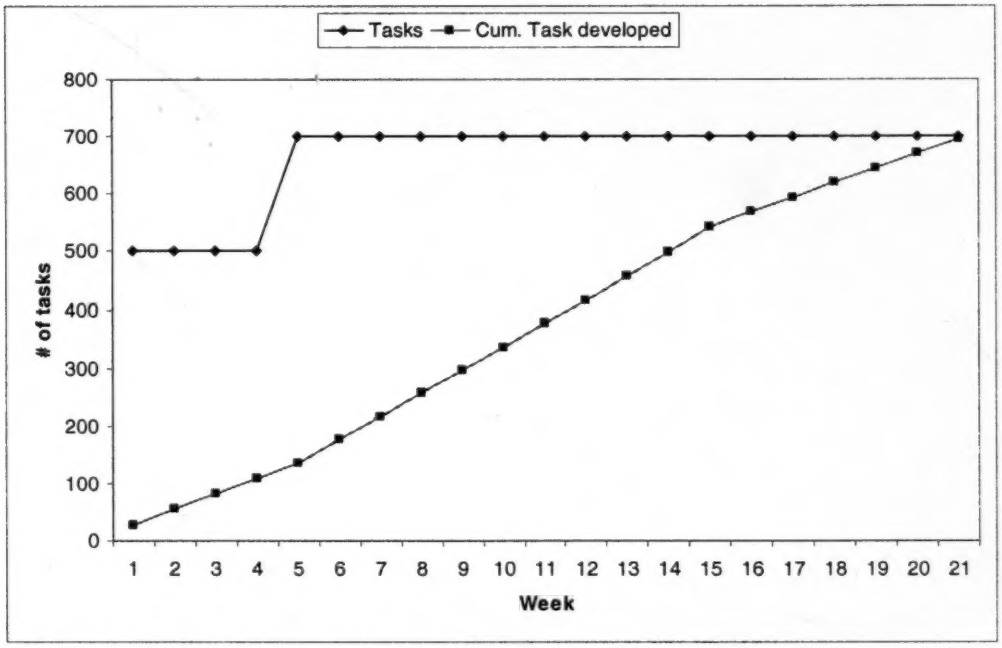


Figure 4.9: Cumulative Tasks Developed & Required Tasks

Figure 4.9 shows the respective cumulative tasks developed for each week, and the required number of tasks to be completed in the project. The project size was increased 40% after the 4th week of development, showing a sharp increase from 500 function points to 700 function points. It is assumed here the project size could be

adjusted within one week, i.e., during the 5th week. After the new size was introduced, the cumulative tasks developed was increased slightly faster than before from Week 6 to Week 15, shown by a steeper slope in the graph. This is due to a higher productivity from the employees' overtime hours. However, at the end of 15th week, the increase rate of cumulative tasks developed dropped slightly due to reaching the employees' overwork duration threshold, which is 10 weeks. Hence, after 10 weeks of overtime, the workers were no longer willing to continuously working over the hours and their productivity returned to the original value. Nevertheless, from the cumulative tasks developed line, it shows that by Week 21, it reaches the 700 mark. Hence, the development team finally managed to pull off this overtime strategy while maintaining the original schedule.

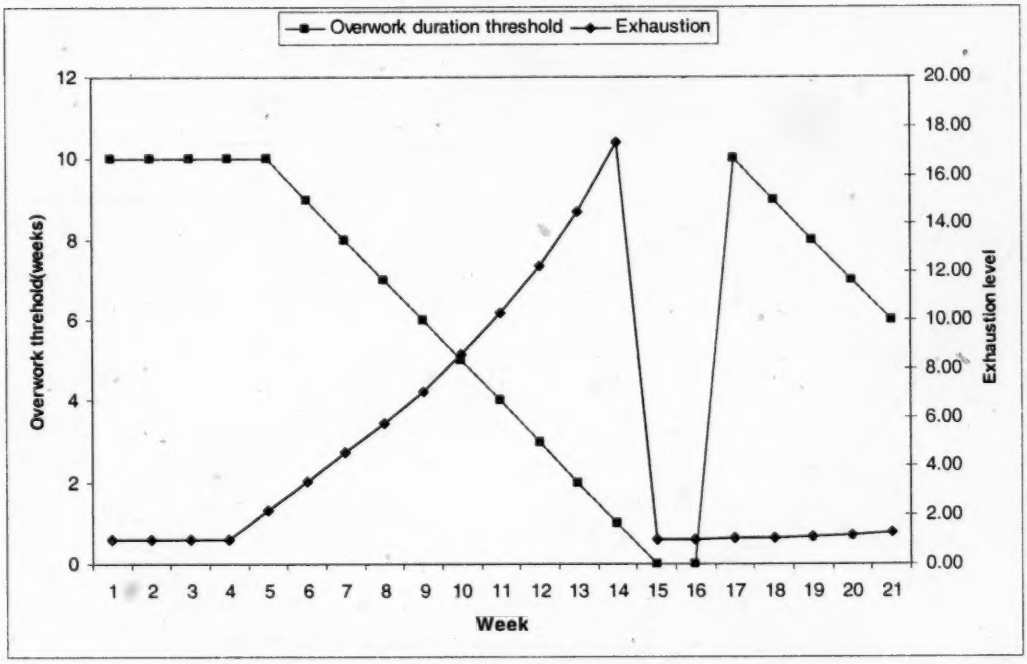


Figure 4.10: Exhaustion Level & Overwork Duration Threshold

Figure 4.10 shows the workers' overall exhaustion level and the depletion of the overwork duration that the workers are willing to handle. The exhaustion level starts at 1 initially, and this value is chosen arbitrary. The rate of increase in exhaustion level is based on the graph in Figure 3.9. When the size was increased by 40%, the

employees were required to work overtime, and the exhaustion level started to increase from Week 5. Since the overwork duration threshold was 10 weeks, the employees were only willing to work overtime from the beginning of Week 5 till the end of Week 14, which is marked by an exponential increase in exhaustion level. When the threshold was used up, the employees needed two weeks of de-exhaustion or recuperating period before a new cycle of 10-week overwork threshold was set. Hence, the exhaustion level drops back to 1 during that 2-week period. After the recuperation period was over, the employees started to work overtime again, but worked less overtime as compared to the first cycle. The reason was most of the tasks had been completed and they were no longer needed to work as hard.

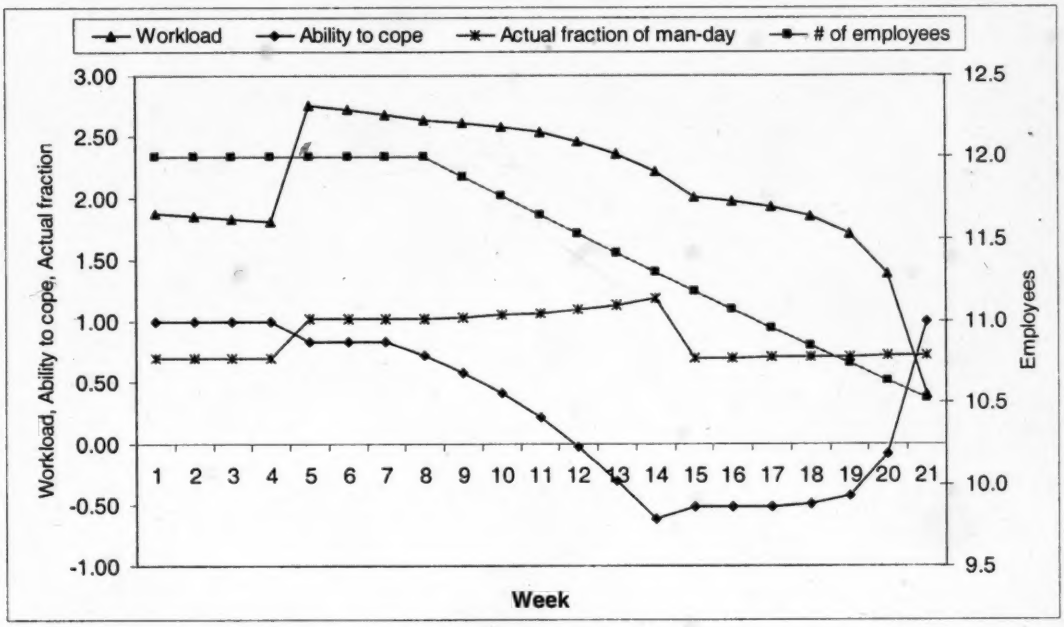


Figure 4.11: Workload, Ability to cope, Actual fraction of a man-day & Number of employees

Figure 4.11 shows a combination of work related parameters. Initial value of ability to cope is set at 1, which is chosen arbitrary. The ability to cope measures the level of workers' ability to handle their job, and is dependent on both workload and exhaustion. Exhaustion level has been explained before, while workload is a function of number of remaining tasks being distributed over the project workforce within the

remaining project development time. The decreasing and increasing rate of ability to cope are based on Figure 3.8. Based on the above figure, when the size was increased by 40%, the ability to cope started to drop, mainly due to the increased on exhaustion level and workload. The workload was increased as more tasks were needed to be completed by the same number of employees, as the policy was not to hire new employees. For the next two weeks, Week 6 and Week 7, the ability to cope stabilized as the exhaustion level at that time was within the employees' tolerance level. Hence, even they were slightly exhausted, it didn't affect their ability to cope. However, starting Week 8, the exhaustion exceeded the level the employees can handle and resulted in a sharp drop in their ability. By Week 15, the decline in the workers' ability to cope was halted. The reason is the exhaustion level returned to normal when the workers had their chance to recuperate for two weeks. By the end of the last two weeks of project development, the ability to cope increased as the workload dropped further when the project was about to be completed.

The actual fraction of a man-day shows the fraction of working hours in a day spent on this project. Initially, when the size had not been changed, the actual fraction was close to the nominal fraction, which was 0.7. However, once the increased size was released in the development team, the actual fraction increased by 45% starting Week 5, when overtime was needed. As a result, the productivity was increased by an equivalent amount. When the workers stopped working overtime at Week 15, the actual fraction dropped back to the nominal value

The number of employees working in this project exhibits an interesting trend. This number is displayed in fraction, although not possible in reality, in order to show a gradual decline in manpower. The workers were required to give a 4-week resignation notice. It is assumed that when the ability to cope level dropped below 1, the workers had a slight tendency to quit (1% quitting rate), and when the level dropped below –1, they were more likely to resign (2% quitting rate). This range is chosen arbitrary and may vary in different organizations. As the quitting rate is dependent on the ability to cope, it is observed that at Week 5, the ability dropped

below 1. This resulted in a slight increase in quitting tendency, and 1% of the employees were going to quit. The effect was realized 4 weeks later when they actually left the development team. Thus, in Week 9, we can see a decline in number of employees. As the ability to cope stayed below 1 throughout the project development, there was always a slight tendency to quit, and thus we can see a constant decline.

Workload depends on remaining tasks, remaining development time, and project workforce. For the first 4 weeks, when the size had not been increased, we can see a constant drop in workload as more tasks were completed. Nevertheless, when the 40% size increment was introduced in Week 5, the workload increased by an equivalent amount, and then dropped gradually as more tasks were being developed. When all tasks were finally completed, the workload became zero.

The above three scenarios are just some examples of the modeling application. There are various possible cases. For example, both of the project duration and the manpower sought can be varied simultaneously with some overtime to determine the solution suitable for the project manager. The organization may not want to hire all 4 engineers in order to meet the original deadline or continuously drive its existing employees to overwork for 10 weeks. Instead, they can afford to hire fewer engineers by compromising the project deadline and without driving the worker to full exhaustion. Hence, different scenarios can be simulated in this integrated multi-project management model, but in order to realize its full potential, simulation software should be used, as discussed in the next section.

5.0 FUTURE RESEARCH

To further evaluate and expand the domain of this software multi-project management, the following proposals are suggested for future investigation.

5.1 MODEL SIMULATION

Upon examining the system dynamics models, it is apparent that the multi-project environment in software engineering involves dynamic complexity. Simulation using the system dynamics models may provide a solution and an application to this approach as previously described in Section 2.3.2. The software that provides the simulation environment must be equipped with the following capabilities:

- CCPM project network construction feature as a foundation for the multi-project environment.
- User scenario interface that provides feasibility to manipulate and control certain scenario elements.
- User output interface that displays crucial results during and upon completion of simulation.
- User input interface that allows managers to supply different system dynamics models that are deemed appropriate, and the ability to manipulate the model parameters.
- System dynamics model platform that provides the core medium for dynamic simulation that predicts how a system evolves and responds to its environment.
- Some degree of separation between shared models and individual project models, which supports the distinction between these two types.
- Database that stores the project network, models, parameters and results.
- Controller as an engine of integration across multiple models and platform.

Using this advanced simulation software package, the models can be built into the software and different scenarios can be tested. Different scenarios arise during project development due to different events, policies, and strategies. A project manager can

plan for the expected behavior of a project development process but the unexpected scenarios, such as a spike in attrition rate due to better job opportunity elsewhere, may affect the project development in terms of schedule and costs. Hence, some degree of uncertainties must be considered in the simulation as some of the possible scenarios. By testing different scenarios, the manager may understand the effects with several combinations of events on the process. By capturing different scenarios in a model, the managers are able to build a reusable knowledge base for the project management.

5.2 Resource Allocation Heuristics

The modified Wiest and Levy resource leveling heuristics (Walker 2001) mentioned in this paper provides a fundamental and feasible approach to allocate resources in a multi-project environment. However, there are a few factors that need to be incorporated to customize the heuristics and the dynamic model into software engineering.

- There is a lack of using project priority in the resource allocation heuristics. The only factor used in the current heuristics to allocate the resources is the slack time. In the software organization, a low priority project may have slack time of zero in some activities, while a higher priority project may have non-zero slack time in some activities. Hence, if there is a resource contention between these two projects at some point in time, the current heuristics will allocate resource to the low priority project since the slack time is zero. Further improvement can be made by adding the priority factor in the heuristics.
- Based on the Theory of Constraints and the bottlenecks illustrated in the job shop environment in one of the Goldratt's book (Goldratt 1992), bottlenecks/constraints are machines, processes or resources that limit the organization from achieving its goal. In software engineering, key personnel that have specific valuable knowledge are in high demand across multiple projects, and often enough are the bottlenecks that restrict the project

development. To apply the Theory of Constraints, we follow the following steps (Goldratt 1992):

i. Identify the constrained resources: Resources that are deemed crucial should be segregated from the general resource group, and are identified as a separate resource group.

ii. Increase constrained resources efficiency: When allocating resources across various projects, the constrained resources should always have the highest priority being allocated to the project activities to ensure minimum personnel's slack time. The key personnel should not stay idle while waiting for another project. In addition, the key resources should only be allocated on activities that have been validated and verified to minimize unnecessary rework. For example, if a customer has not finalized the software specifications and requirements, the key personnel that will work on the software design and architecture should not be assigned to work on this project yet to avoid unnecessary alterations.

iii. Subordinate everything else to the above decision in Step ii. Since the subsequent work depends on the key resources, the management should apply all available resources to assist in breaking them, or in this case, train the non-key personnel with specific skills. In practically all cases, their limiting impact can be reduced or eliminated.

iv. Elevate the constraints. If we continue to work toward breaking a constraint, at some point the constraint will no longer be a constraint. The constraint will be broken.

v. If the constrained resources are no longer critical, return to Step i. When that happens, there will be another constraint that is limiting the progress to the goal.

The above first two steps (Steps i and ii) can be included in the resource allocation heuristics, i.e., a separate resource group is identified and assigned for the constrained resources and the management should ensure the peak efficiency of those personnel.

- In software engineering, immediate job transferring from one project to another will require some additional time for the people to acclimatize to the requirements and design of the new project. By transferring the resource from one project to another, it may prolong the duration of that activity. Hence, this can be taken into account in the simulation model by adding a delay factor.

6.0 CONCLUSIONS

The software industry commonly works in multi-project environment, which may lead to issues of resource sharing, variations in projects' completion date, unexpected events, changes in human resource policies and other related interdependency issues. For example, adding another project to the network or making changes in an existing project will affect the productivity and availability of resources currently employed and consequently the completion time of other existing projects. It may be impossible to visualize the complexity of this system without using systems thinking approach. We study the system by understanding the collective behavior and connectedness of its components. The cause of an event results in an effect on another component, and the chain of reaction may go on throughout the entire system. Hence, as a start, we need to look at the system as a whole, and understand that one small variation in one component may propagate widely. Subsequent effort requires dividing the system into sub-models, represented by causal loop diagrams and flow diagrams that depict the continuous flow of information.

The system dynamics models in this analysis involve 6 sub-models. Two of which are Shared Models. These models are constituted by variables and factors that are shared among various projects running simultaneously and have mutual effects on all projects. The remaining four models have their characteristics that are unique to each individual project. These Individual Project models do not interact and share their cause-and-effect relationship with other projects.

The composition of these models, i.e., Workload and Exhaustion model, Human Resource model, Effort model, Productivity model, Control model and Planning model, offers a window to study the complex phenomena involving interdependencies in a software multi-project environment. The relationships may not be applicable to all organizations, but they may provide an insight to how this complex system can be analytically studied and accustomed to individual scenarios.

Scenario planning is an extension of the systems thinking paradigm. It is utilized and constructed in this environment by manipulating various system dynamics parameters in the models. Each scenario, as a result of different settings, represents a distinct, conceivable world that may lead to diverse consequences. Scenarios represent events, policies, theories, and strategies that are practices that may impose on the project. The project elements that belong to those four categories need to be identified. Computer simulation is then used to execute diverse scenario cases, as different situations may arise throughout the project development. Hence, systems thinking with scenario modeling is suitable to dynamically construct different plausible paths, and the results may help the management to prepare the uncertainties in a structure way.

The multi-project environment resides on a foundation constructed using a CCPM network. CCPM involves resource allocation and leveling across multi-projects and buffer placement within critical chain. It uses a common method to identify the critical chain in a multi-project network, and recognizes the interdependencies of various projects in resource requirements, and accommodates the nature of individual projects in resource allocation. By providing a whole system view of the projects, it identifies and protects what's critical from inevitable uncertainty. In other words, CCPM helps the manager to recognize the project paths that are crucial in ensuring the on-time delivery of the projects.

Integrating the CCPM network, the scenario model, and the system dynamics models with a controller, serves the purpose of managing a multi-project organization in a more predictive and controllable manner. The scenario model describes various potential cases that may happen during project development. The CCPM network then shows the interconnection of the projects in resource allocation, activity sequencing and completion date. The dynamic models utilize the information supplied by the scenarios and the multi-project network to model feedback loops needed to describe cause-and-effect relationships of the development environment. The controller acts as a medium and regulator for information and command passing. Hence, each unit provides its own functions in harmony.

The results of the simulation can be used to study various interactions within the multi-project environment, and to build a reusable knowledge base for future reference. It provides a means to study complex phenomena in project development that cannot be carried out easily with actual cases. The model allows the project managers to study and assess different effects of causal relationships influencing various concurrent projects, and can offer them insights into the outcomes of different management approaches.

This concept of integration, however, requires next generation simulation software to realize the full potential in order to sort out the best solutions to produce a well-rounded multi-project simulation model. Nevertheless, the main contribution of this research is to show how to integrate and use these models to study the multi-project environment. Different organizations may employ different strategies and consist of more complicated relationships among their models. Regardless of their complexity, all models by their very nature are not identical to the reality, but the systems thinking may provide a powerful perspective of reflecting a sense of truth in a complex system.

BIBLIOGRAPHY

Abdel-Hamid TK, Madnick SE. 1983. An Integrative Approach to Modeling the Software Management Process: A Basis for Identifying Problems and Evaluating Tools and Techniques. *IEEE Computer Society Workshop on Software Engineering Technology Transfer*, Miami Beach, Florida.

Abdel-Hamid TK, Madnick SE. 1991. *Software Project Dynamics: An Integrated Approach*. Prentice Hall, Englewood Cliffs, New Jersey: 1-264.

Artzer SP, Neidrauer RA. 1982. Software Engineering Basics: A Primer for the Project Manager. *Naval Postgraduate* School, Monterey, California.

Baddoo N. 2002. Motivators of Software Process Improvement: An Analysis of Practitioners' Views. *The Journal of Systems and Software* 62(2): 85-96.

Barros M, Werner CML, Travassos GH. 2000. Applying System Dynamics to Scenario Based Software Project Management. *Proceedings of the 2000 International System Dynamics Conference*, Berghen.

Barros M, Werner CML., Travassos GH. 2000. Using Process Modeling and Dynamic Simulation to Support Software Process Quality Management. *XIV Simpósio Brasileiro de Engenharia de Software, Workshop de Qualidade de Software*, João Pessoa, Brazil.

Brooke B. 1996. Studies label training costly, cost effective. *Philadelphia Business Journal,* http://www.bizjournals.com/philadelphia/stories/1996/11/04/focus3.html, January 21 2004.

Brooks FP. 1995. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley (anniversary edition), Sydney, Australia: 1-322.

Boehm BW. 1981. *Software Engineering Economics*. Prentice Hall Inc., Englewood Cliffs, New Jersey: 1-767.

Boehm BW, Clark B, Horowitz E, Westland C, Madachy R, Selby R. 1995. Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering, Science Publisher,* Amsterdam, Netherlands: 57-94.

Booth AL, Chen YF, Zoega G. 2002. Hiring and Firing: A Tale of Two Thresholds. *Journal of Labor Economics* 20(2): 217-248.

Carayannis ES., Kwak YH., Anbari FT. 2003. The Story of Managing Projects: A Global, Cross-Disciplinary Collection of Perspectives, Greenwood Press.

Churchman CW, Ackoff RL, Arnoff L. 1957. Introduction to Operations Research. *John Wiley and Sons,* New York: 1-645.

Coyle RG. 1996. *System Dynamics Modelling – A Practical Approach*. Chapman & Hall, London: 1-413.

Davidson EJ. 1999. Joint Application Design in Practice. *Journal of Systems and Software* 45(3): 215-223.

Donzelli P, Iazeolla G. 2001. Hybrid Simulation Modelling of the Software Process. *The Journal of Systems and Software* 59(3): 227-235.

Dye LD, Pennypacker JS. 2000. Project Portfolio Management and Managing Multiple Projects: Two Sides of the Same Coin?. *Proceedings of the Project Management Institute Annual Seminars & Symposium,* Houston, Texas.

Fatemi-Ghomi S.M.T., Ashjari B. 2002. A Simulation Model for Multi-Project Resource Allocation. *International Journal of Project Management* 20(2): 127-130.

Ford D. 1995. The Dynamics of Project Management: An Investigation of the Impacts of Project Process and Coordination on Performance. *MIT PhD Dissertation in Dynamic Engineering Systems*, Cambridge, Massachusetts.

Forrester JW. 1961. *Industrial Dynamics*. The MIT Press, Cambridge, Massachusetts: 1-464.

Garmus D, Herron D. 2002. Estimating Software Earlier and More Accurately. *CrossTalk: The Journal of Defense Software Engineering*, http://www.stsc.hill.af.mil/crosstalk/2002/06/garmusherron.html, July 13, 2003.

Gewirtz ML, Lindsey A. 2000. Women in the new Economy: Insights and Realities. *WorldWIT*, http://www.worldwit.org/features/women_economy_survey.pdf, July 11, 2003.

Goldratt EM. 1997. *Critical Chain*. North River Press Publishing Corporation, Great Barrington, MA: 1-246.

Goldratt EM, Cox J. 1992. *The Goal: A Process of Ongoing Improvement.* North River Press Publishing Corporation, Croton-on-Hudson, NY: 1-337.

Hanson D. 2000. Training increases profits, reduces turnover. *Organizational Developments* 12(3): 1-4.

Helpman E, Rangel A. 1999. Adjusting to a New Technology: Experience and Training. *Journal of Economic Growth* 4: 359-383.

Henderson L. 2001. The choice for more training. *CenterWatch* 8(3): 1-6.

Hendriks MHA, Voeten B, Kroep L. 1999. Human resource allocation in a multi-project R&D environment. Resource capacity allocation and project portfolio planning in practice. *International Journal of Project Management* 17(3): 181-188.

Jeffery DR, Scott L. 2002. Has twenty-five years of empirical software engineering made a difference. *Proceedings Asia-Pacific Software Engineering Conference*: 539-546.

Kirkwood CW. 1998. System Dynamics Methods: A Quick Introduction. *Department of Management-Arizona State University*, http://www.public.asu.edu/~kirkwood/sysdyn/SDIntro/SDIntro.htm, July 13, 2003.

Koch MJ, McGrath RG. 1996. Improving labor productivity. Human resource management policies do matter. *Strategic Management Journal* 17(5): 335-354.

Lakhanpal B. 1993. Understanding the Factors Influencing the Performance of Software Development Groups: An Exploratory Group-Level Analysis. *Information and Software Technology* 35(8): 468-471.

Lehman MM, Ramil JF, Kahen G. 2001. A Paradigm for the Behavioural Modelling of Software Processes using System Dynamics. *Dept of Computing, Imperial College – Technical Report*: 1-11.

Lehman MM, Wernick PD. 1998. System Dynamics Models of Software Evolution Processes. *Proc. ICSE98 Int. Workshop on Princ. of Soft. Evolution - IWPSE98*, Kyoto, Japan: 6-10.

Levary RR, Synott DJ, Lin CY. 1988. An Intelligent Dynamic Simulation Model for Designing Software Development Processes. *Omega* 16(6): 569-575.

Linberg KR. 1999. Software Developer Perceptions About Software Project Failure: A Case Study. *The Journal of Systems and Software* 49(2): 177-192.

Long DD. 1995. Coping with Communication Overhead: The New Productivity Crisis?. *Ernst & Young Center for Business Innovation* Working Paper CBI306, Boston, Massachusetts.

Madachy R. 1996. System Dynamics Modeling of an Inspection-Based Process. *Proc. of the Eighteenth International Conference on Software Engineering*: 376-386.

Mahaney RC, Lederer AL. 2003. Information Systems Project Management: An Agency Theory Interpretation. *Journal of Systems and Software* 68(1):1-9.

Maurer F, Martel S. 2002. Extreme Programming: Rapid Development for Web-based Applications. *IEEE-Internet Computing* 6(1): 86-90.

Merrill D, Collofello JS. 1997. Improving Software Project Management Skills Using a Software Project Simulator. *Frontiers In Education Conference 1997 (FIE97*: Pittsburgh, Pennsylvania.

Miranda E. 2002. Planning and Executing Time-Bound Projects. *Computer* 35(3):73-79.

Payne JH. 1995. Management of multiple, simultaneous projects: a state of the art review. *International Journal of Project Management* 13(3): 163-168.

Pfahl D, Klemm M, Ruhe G. 2000. Using System Dynamics Simulation Models for Software Project Management Education and Training. *IESE-Report 035.00/E*: 1-5.

Pfahl D, Lebsanft K. 2000. Using simulation to analyse the impact of software requirement volatility on project performance. *Information and Software Technology* 42(14): 1001-1008.

Pressman RS. 2001. *Software Engineering A Practitioner's Approach*. McGraw Hill, New York City, New York: 1-860.

Punie Y, Burgelman JC, Bogdanowicz M, Desruelle P. 2001. The Future of News Media Industries: Scenarios for 2005 and Beyond. *Multimedia Developments in the Information Age Report.* http://www.ecdc.nl/publications/index.php?pub_type=1, December 10, 2003.

Reifer DJ. 2002. How Good are Agile Methods. *IEEE-Software* 19(4): 16-18.

Ruiz AC., Gomez MLN. 2002. Factors affecting quits and layoffs in Spain. *Fundación Centro de Estudios Andaluces: Economic Working Papers at centrA*, E2002/16, Sevilla, Spain.

Sharp HC, Woodman M, Hovendon F, Robinson H. 1999. The Role of Culture in Successful Software Process Improvement. *Proceedings of the 25th Euromicro Conference*: 170-176.

Sharpe A. 1999. Organizational Structure, Information Technology and Productivity: Can Organizational Change Resolve the Productivity Paradox?. *Human Resources Development Canada* , R-98-6E, Canada.

Sherwood D. 2002. *Seeing the forest for the trees: A manager's guide to Applying Systems Thinking*. Nicholas Brealey Publishing, London, England: 1-240.

Sterman JD. 1992. System Dynamics Modeling for Project Management. *Technical Report MIT System Dynamics Group*, Cambridge, Massachusetts.

Teasley SD, Covi LA, Krishnan MS, Olson JS. 2002. Rapid Software Development through Team Collocation. *IEEE Transactions on Software Engineering* 28(7): 671-683.

Turner JR. 1993. The Handbook of Project-based Management. *McGraw-Hill,* London.

Turner JR, Payne JH. 1997. The problem of projects of differing size and skill mix. *Project Management: the Journal of the Project Management Association* 3(1): 14-17.

Walker II ED. 2000. An Introduction to Critical Chain Project Management. *Proc.of Southeast Decision Science Institute*, Wilmington, North Carolina: 205-207.

Walker II ED. 2001. Towards a More Effective Method of Scheduling Resource-Constrained Multiple Projects. *Proc. of the Twelfth Annual Conference of the Production and Operations Management Society,* Orlando, Florida.

Warren C. 2003. Workplace health – national survey results. *Media Entertainment and Arts Alliance*, http://www.alliance.org.au/images/2003/news.htm, July 13, 2003.

Warren K. 2002. *Competitive Strategy Dynamics*. John Wiley & Sons Ltd., West Sussex, England: 1-346.

Wendorff P. 2002. Systems Thinking in Extreme Programming. *European Conference On Information Systems 2002,* Gdañsk, Poland.

White KS. 1999. Software Engineering Management for Productivity and Quality. *International Conference on Accelerator and Large Experimental Physics Control Systems*, Trieste, Italy.

Wiest JD, Levy FK. 1977. A Management Guide to PERT/CPM with GERT/PDM/DCPM and Other Networks. *Prentice Hall* 2nd edition, Englewood Cliffs, New Jersey.

Williams LA, Kessler RR. 2000. All I Really Need to Know about Pair Programming I Learned in Kindergarten. *Communications of the ACM* 43(5): 109-114.

APPENDIX

Week	Tasks	Num of employees	Workload	Ability to cope	Actual productivity	Cum. Task developed	Man-wk remaining	Overwork duration threshold	%boost in workrate	Actual fraction of man-day	Exhaustion	Tendency of quitting	Weekly quitting rate %
1	500	12.0	1.88	1.00	2.2840	27.4	240.00	10	0.00	0.70	1.00	None	0
2	500	12.0	1.85	1.00	2.2840	54.8	228.00	10	0.00	0.70	1.00	None	0
3	500	12.0	1.83	1.00	2.2840	82.2	216.00	10	0.00	0.70	1.00	None	0
4	500	12.0	1.81	1.00	2.2840	109.6	204.00	10	0.00	0.70	1.00	None	0
5	700	12.0	2.76	0.83	2.2840	137.0	192.00	10	0.45	1.02	2.18	May be	1
6	700	12.0	2.72	0.83	3.3209	176.9	180.00	9	0.45	1.02	3.36	May be	1
7	700	12.0	2.68	0.83	3.3209	216.7	168.00	8	0.45	1.02	4.53	May be	1
8	700	12.0	2.64	0.72	3.3209	256.6	156.00	7	0.45	1.02	5.71	May be	1
9	700	11.9	2.62	0.58	3.3209	296.0	142.56	6	0.48	1.03	7.05	May be	1
10	700	11.8	2.58	0.41	3.3788	335.8	129.37	5	0.50	1.05	8.56	May be	1
11	700	11.6	2.53	0.21	3.4425	375.9	116.44	4	0.53	1.07	10.27	May be	1
12	700	11.5	2.46	-0.03	3.5145	416.4	103.74	3	0.56	1.10	12.22	May be	1
13	700	11.4	2.36	-0.30	3.6001	457.5	91.30	2	0.61	1.13	14.50	May be	1
14	700	11.3	2.22	-0.61	3.7125	499.4	79.08	1	0.69	1.18	17.34	May be	1
15	700	11.2	2.00	-0.52	3.9038	543.1	67.11	0	0.00	0.70	1.00	May be	1
16	700	11.1	1.98	-0.51	2.3124	568.7	55.36	0	0.00	0.70	1.00	May be	1
17	700	11.0	1.93	-0.51	2.3161	594.1	43.85	10	0.02	0.71	1.04	May be	1
18	700	10.9	1.85	-0.49	2.3582	619.7	32.56	9	0.02	0.71	1.09	May be	1
19	700	10.7	1.70	-0.43	2.3715	645.1	21.49	8	0.02	0.72	1.14	May be	1
20	700	10.6	1.39	-0.09	2.3834	670.5	10.64	7	0.03	0.72	1.21	May be	1
21	700	10.5	0.41	1.00	2.3938	695.7	0.00	6	0.03	0.72	1.28	None	0

Appendix 1: Tabulated results for Scenario 3 Modeling